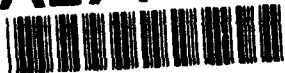


AFIT/GCE/ENG/93D-03

AD-A274 075



DTIC
ELECTE
DEC 27 1993
S A

VULNERABILITY ASSESSMENT
USING A FUZZY LOGIC BASED METHOD

THESIS

Richard W. Fleming, Captain, USAF

AFIT/GCE/ENG/93D-03

This document has been approved
for public release and sale; its
distribution is unlimited.

93-31036



Approved for public release; distribution unlimited

93 12 22 1 49

Disclaimer

The views expressed in this thesis are those of the author and do not reflect official policy or position of the Department of Defense or the U.S. Government.

Accession For	
NTIS <input checked="" type="checkbox"/> DRAP	
DTIC <input checked="" type="checkbox"/> Info	
U.S. <input checked="" type="checkbox"/> Govt	
J. <input checked="" type="checkbox"/> Information	
By	
Distribution	
Availability	
Dist	Availability Special
A-1	

DTIC QUALITY INSPECTED 6

AFIT/GCE/ENG/93D-03

**VULNERABILITY ASSESSMENT
USING A FUZZY LOGIC BASED METHOD**

THESIS

**Presented to the Faculty of the Graduate School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Computer Engineering**

**Richard W. Fleming, B.S.
Captain, USAF**

December 1993

Approved for public release; distribution unlimited

Preface

The purpose of this thesis is to demonstrate the feasibility of using a qualitative analysis method to assess and evaluate computer security vulnerabilities. The primary motivation for this research is to assist the United States Air Force (USAF) in assessing and eliminating the vulnerabilities identified in USAF computer systems. Although the main focus of this thesis is to evaluate computer security vulnerabilities, the methods involved have application in other areas requiring evaluation using qualitative methods.

This thesis puts head-to-head a quantitative approach to analysis and a qualitative approach utilizing linguistic variables. Linguistic variables are represented using a calculus described by Lofti Zadeh (Zadeh65:338). Linguistic variables are terms such as Low, Medium, and High. In the realm of vulnerability analysis, these have a definite semantic meaning. It is proposed, and demonstrated by this thesis, that the use of qualitative analysis using linguistic variables to describe the impact of computer security vulnerabilities is not only feasible, but intrinsically easier to understand and use than quantitative methods.

In developing the necessary computer programs and writing this thesis, I have had a great deal of help from others. I am most indebted to my faculty advisor, Major Gregg Gunsch. His consistent support and guidance was always felt and much appreciated in the many times of need and confusion. I wish to thank Drs. Henry Potoczny and Eugene Santos for serving on my thesis committee. I also wish to acknowledge the Air Force Cryptological Support Center (AFCSC) for their generous sponsorship of this work.

Finally, I wish to thank my wife, Kim, and my children, Dakin and Mandi, for their never-ending support, concern, and love as I spent the many days and nights sequestered in my office with this work.

Richard W. Fleming

Table of Contents

	Page
Preface	ii
List of Figures	vi
List of Tables	vi
Abstract	vii
I Introduction	1
1.1 General Issue	1
1.2 Background	1
1.3 What are Risk and Vulnerability Analysis?.....	4
1.4 Motivation	5
1.5 Hypothesis.....	5
1.6 Research Objectives.....	6
1.7 Scope	6
1.8 Document Structure.....	6
1.9 Summary	7
II Historical Development.....	8
2.1 Introduction.....	8
2.2 Overview of Problem.....	9
2.3 Reasoning with Uncertainty	10
2.3.1 Introduction of AI Methods	10
2.3.2 Bayes' Theorem.....	10
2.3.3 Bayesian Networks	12
2.3.4 Dempster-Shafer	13
2.3.5 Fuzzy Set Theory	14
2.4 Analysis Methods.....	17
2.4.1 Introduction	17
2.4.2 Quantitative Analysis.....	19
2.4.3 Qualitative Analysis.....	21
2.5 Organization of Vulnerability Information	22
2.6 Current Air Force Analysis Tools.....	23
2.7 Automating Computer Security Analysis.....	24
2.8 Summary	26
III Methodology.....	27
3.1 Introduction.....	27

3.2 Overview	27
3.3 Background	29
3.4 Statistical Analysis	29
3.5 Fuzzy Logic	31
3.5 Schmucker's Calculation Method	32
3.5.1.1 Translation of Fuzzy Set to Linguistic Term	35
3.5.1.2 Normalization and Convexity of Fuzzy Sets	36
3.5.2 Table Lookup Method	37
3.5.3 Fuzzy Arithmetic	39
3.6 Scalability	44
3.7 Sample Data Generation	44
3.8 Summary	45
IV Implementation	47
4.1 Introduction	47
4.2 Quantitative Method using Statistical Analysis	47
4.3 Qualitative Method using Fuzzy Logic	49
4.3.1 Fuzzy Math	49
4.4 Summary	50
V Results	52
5.1 Introduction	52
5.2 Hypothesis (restated)	52
5.3 Effectiveness	53
5.3.1 Ability to Categorize Vulnerabilities	53
5.3.1.1 Accuracy	53
5.3.1.2 Precision	54
5.3.1.3 Generation of Distinctions	56
5.3.1.4 Evaluation of Cose Results	57
5.3.2 Ease of Use	58
5.3.2.1 Interpretation of Data	58
5.4 Efficiency	61
5.4.1 Timed Performance	61
5.4.2 Scalability	61
5.5 Summary	61
VI Conclusions and Recommendations	63
6.1 Conclusions	63
6.2 Project Recommendations	63
6.3 Future Enhancements and Phases	63
Appendix A: Computer Programs	66
A-1 Quantitative Method using Statistical Analysis	66
A-2 Qualitative Method using Fuzzy Analysis (Table Lookup)	70

Appendix B: Sample Program Output.....	75
B-1 Program Output - Numeric Method (50 Samples)	75
B-2 Program Output - Fuzzy Method (50 Samples)	81
Bibliography.....	88
VITA	91

List of Figures

Figure	Page
1. Fuzzy Sets Describing Height.	15
2. Loss Unit Concept.....	19
3. Defined Fuzzy Sets.....	39
4. Positive Increasing Fuzzy Sets.	42
5. Median Based Fuzzy Sets.	42
6. Relative Magnitude Fuzzy Sets.	44
7. Lisp Object with Numerical Data	46
8. Lisp Object with Fuzzy Data.....	46
9. CLOS Object Structure	48
10. Clustering of Audit Functional Area (Quantitative)	54
11. Clustering of Audit Functional Area (Qualitative)	54
12. Timed performance (Numerical vs. Fuzzy).....	62

List of Tables

Table	Page
1. Addition of Fuzzy Sets (Linear Mapping)	51
2. Multiplication of Fuzzy Sets (Linear Mapping)	51
3. Timed performance (Numerical vs. Fuzzy).....	62

Abstract

This thesis demonstrates the feasibility of using qualitative analysis methods to evaluate computer security vulnerabilities. Although many risk analysis systems exist, few provide for the adequate analysis of identified vulnerabilities. While the main focus of this thesis is to evaluate computer security vulnerabilities, the methods involved have application in other areas requiring evaluation using qualitative methods.

It is proposed, and demonstrated by this thesis, that the use of qualitative analysis using linguistic variables to describe the impact of computer security vulnerabilities is not only feasible, but intrinsically easier to understand and use than quantitative methods.

VULNERABILITY ANALYSIS

USING A FUZZY LOGIC BASED METHOD

I Introduction

1.1 General Issue

This thesis describes, develops, and compares automated analysis methods providing support capabilities to security analysts in the evaluation of computer security vulnerabilities.

1.2 Background

Computer security is a major Air Force concern. Air Force agencies and organizations use computers in almost every aspect of their operations. To maintain even minimal operational capability, organizations must increase their dependence on these machines. Associated with this increased dependence is an increase in associated costs, both tangible and intangible, resulting from compromised computer resources. Compromised resources occur as the result of vulnerabilities being exploited. These vulnerabilities include, but are not limited to, unauthorized access, sabotage, physical damage, and accidental misuse.

To ensure the protection of computer resources, the USAF implemented AFR 205-16, Security: Computer Security Policy. The purpose of this regulation is to ". . . protect the confidentiality, integrity, and availability of information processed on all Air Force computer systems" (DAF89:1). Air Force Systems Security Instruction (AFSSI) 5100, The Air Force COMPUSEC Program and AFSSI 5102, Computer Security for Operational Systems later

supplemented AFR 205-16. AFSSI 5102 requires each facility processing information for the USAF to perform a security risk analysis on each installed computer. This requirement includes both government- and contractor-owned facilities (DAF93b:2). This requirement to perform a security risk analysis applies not only to Air Force systems, but as directed by the Office of Management and Budget of the United States Circular Number A-71, to ". . . every federal department or agency operating one or more computer installations . . ." (Carroll84:2)

Risk analysis identifies threats and vulnerabilities associated with a given computer system and determines if the safeguards in effect adequately protect the system from compromise. After eliminating the risk, or reducing the risk to acceptable levels, the computer security officer (CSO) authorizes the computer system to be operated at a maximum sensitivity level and for a certain mission. Any change to the computer system, whether hardware, software, or mission, mandates performing another risk analysis.

When the Air Force established AFR 205-16, most of the computers in the Air Force were large single-site mainframes. Although very complicated and large systems, the configuration and intended usage of these mainframes did not change often. CSOs could manage the required number of risk analyses manually. With the advent of personal computers and desktop workstations, the required number of analyses became too difficult to manage. AFCSC, located at Kelly AFB, TX, developed the Automated Risk Evaluation System (ARES) to address this management problem.

ARES is a computer program that assists computer security personnel in performing a risk analysis. The computer program guides the user through a myriad of questions pertaining to all aspects of computer security. After the user answers all of the pertinent questions, ARES generates a series of reports identifying the possible vulnerabilities associated with a particular computer system.

The listing of identified vulnerabilities generated by ARES does not provide any indication as to the importance of each vulnerability. The CSO must still evaluate the importance of each vulnerability. The manual process of evaluating computer security vulnerabilities is very labor intensive. To help ease this workload, this thesis presents two automated methods possibly useful in the evaluation of computer security vulnerabilities.

At this point, I need to stress a few items. First, regardless of the method used the evaluation of vulnerabilities by a CSO is subjective. Security analysis is not an absolute and the degree of importance assigned to a vulnerability may differ from CSO to CSO.

Second, the evaluation of vulnerabilities is context sensitive with regard to location, hardware, software, and mission requirements. This implies that one site might identify a vulnerability as trivial while another site might identify the same vulnerability as critical. Lack of backup power is an example of a context sensitive vulnerability. Not having backup power for an air traffic control system would probably be critical while not having backup power for electronic message system might be trivial. Even in this example, the use of the system is context sensitive. If the air traffic control system is for a small airport and the airport only uses the system as a secondary method of air traffic control, the system may not be critical to flight operations, hence, the system may not require backup power. Likewise, the vulnerability of not having backup power is critical if world leaders use the electronic message system for communication. This implies that the automated method must allow for each CSO to tailor the importance of vulnerabilities to meet site specific requirements.

Last, and most important, is that there is no proven and demonstrable "industry standard" method for analyzing computer security vulnerabilities. As such, the methods discussed and developed in this thesis cannot be proven to be any better or worse than any other method. This thesis does demonstrate that the methods presented have merit with regard to their ability to

analyze computer security vulnerabilities. Their applicability to a specific computer site is subject to that site's requirements and existing analysis techniques.

1.3 What are Risk and Vulnerability Analysis?

Before continuing, it is essential to define the differences between risk and vulnerability analysis. *Risk* is defined as "the possibility of loss" (Carroll84:xv) while *vulnerability* is defined as "a weakness or lack of controls that would allow or facilitate a threat actuation against a specific asset or target" (Podell86:88). How these terms differ is best demonstrated by an example. The computer system is identified as not having a password capability. This is a vulnerability. The loss or compromise of data *because* the system doesn't have a password capability is a risk. In other words, *risks are caused by a vulnerability being exploited*.

A key component of any risk analysis is *vulnerability identification and analysis* (Carroll84:137). Vulnerability analysis is the evaluation of identified vulnerabilities to determine the importance or impact of each vulnerability with respect to all other identified vulnerabilities. The importance or impact is usually with regard to confidentiality, availability, and integrity of the system. In vulnerability analysis, countermeasures are not considered and *no evaluation is performed with regard to expected loss* (Carroll84:90). In other words, the end product of a vulnerability analysis is the categorization and clustering of vulnerabilities by importance or impact.

Risk analysis is the "analysis of system assets *and vulnerabilities* to determine the system's exposure or expected loss" (Podell86:84). In order to perform an effective risk analysis, a complete vulnerability analysis must be performed. The categorization of the vulnerabilities performed by the vulnerability analysis and the cost information associated with each vulnerability

being exploited are combined and analyzed in a risk analysis. The final product of the risk analysis indicates which clustering of vulnerabilities and cost provide the greatest exposure or expected loss. Risk can only be effectively analyzed if all vulnerabilities are identified and evaluated.

1.4 Motivation

The primary motivation for this research is a need to automate and standardize computer vulnerability analysis. A secondary motivation arose as a result of discussions with the managers of ARES. As with any on-going software project, ARES is under constant modification and revision. One of the revisions planned for ARES is the incorporation of a risk analysis methodology. As proposed by the developers and maintainers of ARES, this risk analysis methodology would use a quantitative analysis approach (Trident93:8). To ensure the best product is fielded, I proposed to the managers of ARES that a qualitative analysis approach might be more intuitive for the end-user to understand and utilize, while still maintaining the effectiveness and efficiency of a quantitative method.

This research is the first phase of a multi-phase research project. This thesis specifically addresses the feasibility of using a qualitative versus quantitative analysis method to evaluate identified vulnerabilities. Future phases outlined in chapter 6 will address other risk analysis capabilities and implementation details.

1.5 Hypothesis

The hypothesis of this thesis is stated in two parts: that a qualitative analysis approach to vulnerability analysis is as effective and efficient as a quantitative approach and that the qualitative

approach provides the security analyst with intuitive information not readily available in quantitative approaches. Effectiveness is measured as the ability to provide reasonable categorizations of identified vulnerabilities based on importance or impact. Efficiency is measured with regard to processing time and scalability of the method.

1.6 Research Objectives

This thesis has two objectives. First, to identify and develop automated methods that provide a consistent evaluation of vulnerabilities. Second, to evaluate each method for effectiveness and efficiency.

1.7 Scope

The scope of this thesis is limited to identifying, developing, and evaluating two methods capable of evaluating computer security vulnerabilities. The vulnerabilities used as test data in this thesis are a subset of the possible vulnerabilities generated by ARES.

1.8 Document Structure

This thesis contains six chapters and two appendices. The motivation and background for this research is provided in this, the first chapter. Also provided in this chapter are the research objectives. The second chapter provides an overview and discussion of vulnerability analysis methods and various automated analysis methods. The third chapter discusses the methodologies used in this thesis and the justification for using those methods. In the fourth chapter, the details of

how the methodologies were implemented are provided. The fifth chapter presents the results of this research, and the conclusions and recommendations are provided in the sixth chapter.

The first appendix contains a high level overview of the Lisp program code implemented for each of the methods. The second appendix contains sample output generated from two of the implemented methods. Each of the sample runs was generated using the same sample of 50 vulnerabilities. The complete source code and sample data files can be obtained by contacting:

Major Gregg Gunsch (AI Lab)
AFIT / ENG
2950 P Street
Wright-Patterson AFB, OH 45433-7765
ggunsch@afit.af.mil

1.9 Summary

There is a need to automate the analysis of identified computer security vulnerabilities. The Air Force developed ARES, a system capable of identifying vulnerabilities, but not providing any analysis capabilities. Two general categories of analysis methods, quantitative and qualitative, can be used to perform vulnerability analysis. In the field of vulnerability analysis, there is currently no accepted standard method. This thesis hypothesizes that the qualitative methods are comparably effective and efficient in performing this analysis and provide intuitive information to the analyst as compared to quantitative methods. To support this hypothesis, both a numeric and non-numeric analysis method were developed and implemented.

II Historical Development

2.1 Introduction

Beginning with the first ENIAC computer used by the War Department during World War II, there has been a need to ensure the security of computing resources. Since the first computers were very large in size and all components located in one facility, physical security of the system was adequate to protect the resources. With the advent of multi-user systems came the need to incorporate protection mechanisms into the computer's operating system. These protection schemes continued to evolve to encompass aspects of physical, data, user, and environmental security.

As computers and their associated operating environments became increasingly complex, it became proportionally difficult to protect these resources. To assist the security managers in analyzing the risk to their computing resources, several analysis methods were devised. These methods involve using some form of reasoning with uncertainty to assist in the risk analysis.

Reasoning with uncertainty is an essential part of performing risk analysis. If the CSO could eliminate all of the uncertainty in computer security, he could simply look up in a table what safeguard to put in place for each vulnerability. However, all of the uncertainty cannot be eliminated. The introduction of new computer technology introduces new risks. With these new risks come new uncertainties and changes in existing uncertainties. Also, as the dependence on computer technology increases, so do the associated costs of these new risks. Because of these ever changing conditions, the analysis method used by the CSO must have the ability to reason with uncertainty.

2.2 Overview of Problem

There are two basic problems with evaluating computer security vulnerabilities. The first is determining what aspects of computer security vulnerabilities to evaluate. This involves establishing the functional areas that each vulnerability affects. For instance, user IDs affect access control and operating system capabilities. The second problem is then determining how to evaluate the vulnerabilities affecting each functional area in order to provide an overall vulnerability rating for each functional area.

For the purpose of this thesis, I defined seven functional areas that each vulnerability can affect. These are *audit capabilities*, *recovery capabilities*, *access control*, *magnetic media control*, *operating system capabilities*, *configuration control*, and *documentation*. Each vulnerability may affect more than one of these functional areas. These seven areas could be further divided in more detailed areas and probably should be in a fielded implementation. For instance, *access control* could be further divided into hardware protection (e.g., call-back modems) and software protection (e.g., password) control schemes. For the purposes of this thesis, however, I will only demonstrate the capability of the methods to handle these seven functional areas. Each method tested will use the same seven functional areas and vulnerabilities to generate a vulnerability rating for each functional area.

A large problem faced in this thesis was deciding how to evaluate the functional areas and the vulnerabilities affecting each area. My problem arose from deciding which method best reasons with uncertainty. Giarratano and Riley define uncertainty as "... the lack of adequate information to make a decision." (Giarratano:89:185). Uncertainty comes from several sources. First, uncertainty occurs when some or all of the data is unobtainable or missing. Unobtainable

implies the data is not available from any source. Second, uncertainty arises when the expert provides inexact or inconsistent information. Third, uncertainty comes from data that is available, but contains errors (Giarratano89:186-190,221). Last, I would add that uncertainty includes information difficult to quantify such as loss of life or delay in mission.

2.3 Reasoning with Uncertainty

2.3.1 Introduction of AI Methods. As stated earlier, a problem faced in this thesis was deciding which method best performs reasoning with uncertainty. The sciences of Artificial Intelligence (AI) and probability theory have much to offer towards helping solve this problem. Discussed below are several approaches to reasoning with uncertainty. These approaches are by no means the only methods available, but they are representative of the varied 'schools of thought' concerning how to reason with uncertain information. Other methods include Intuitionistic Logic (Martin-Lof82), Multiple Valued Logic (Lukasiewicz67), Variable Value Logic (Michalski75), Variable Precision Logic (Michalski86), Default Logic (Reiter80, Yager87), Temporal Logic (McDermott82), and Decision Trees (Quinlan82) to name a few (Dontas87:2).

2.3.2 Bayes' Theorem. The most well known method to deal with uncertainties is *Bayes' Theorem* (Rich91:231, Giarratano89:204, Bacchus90:67). This theorem, shown in Equation (1), forms the basis for conditional probability theory and allows for calculating the inverse or *a posteriori* probability. If the probability of event A occurring given event B, $P(A|B)$, is known and the probability of event B occurring unconditionally, $P(B)$, is known, Bayes' Theorem allows the calculation of the probability of event B given event A, $P(B|A)$. $P(A|B)$ and $P(B)$ are *a priori* and must be known prior to solving for $P(B|A)$, the *a posteriori* probability.

Bayes' Theorem has one advantage over all of the other methods to be discussed. Bayes' Theorem produces a precise and mathematically provable solution if we know these *a priori* probabilities. Bayes' Theorem has several large drawbacks that make it unfeasible to use. It should be noted that Bayes' Theorem is relatively easy to implement and is theoretically easy to understand, but for most real world problems is very difficult to use. First, in order to calculate the *a posteriori* probability, two *a priori* probabilities, $P(A|B_i)$ and $P(B_i)$, have to be known. For many real world problems, knowing or obtaining these probabilities is difficult, if not impossible (Rich91:233, Dillard91:1). Second, if the problem deals with dependent events, and thus joint probabilities, Bayes' Theorem grows exponentially and becomes computationally intractable (Rich91:233).

$$P(B_i|A) = \frac{P(A|B_i) \cdot P(B_i)}{\sum_{n=1}^k P(A|B_n) \cdot P(B_n)} \quad (1)$$

where: $P(B_i|A)$ = the *a posteriori* probability that event B_i occurs given event A occurs
 $P(A|B_i)$ = the *a priori* probability we will observe event A given event B_i occurs
 $P(B_i)$ = the *a priori* probability event B_i will occur independent of any other event
 k = the number of possible events

(Rich91:232)

As elegant and simple as Bayes' Theorem is to understand and implement, its drawbacks prevent it from being applicable to the vulnerability analysis problem. Its inapplicability arises from the varying number of vulnerabilities to be evaluated and the unavailability of a complete set of *a priori* probabilities. It could be argued that the missing probabilities could simply be generated by the expert based on experience. Even with a complete set of *a priori* probabilities, the implementation still could not overcome the exponential performance characteristics of the theorem.

2.3.3 Bayesian Networks. A variation on the pure Bayes' Theorem is Bayesian Networks. This alternative, developed by Judea Pearl (Pearl88), uses a network structure to model the problem. Pearl hypothesized that instead of representing the problem as one large joint probability distribution as required by Bayes' Theorem, the problem could be broken up into a network of individual nodes. Each node is probabilistically independent of all the other nodes and therefore would not suffer from the exponential growth of a standard Bayes' Theorem approach. Those events that are dependent and therefore must be 'processed' together are represented together within a single node (Rich91:239, Oliver90:387).

Although minimizing the combinatorial effects of a pure Bayes' Theorem solution, Bayesian Networks still require the same *a priori* probabilities. Because of a lack of available data on the probabilities that a particular vulnerability will be exploited, this method is also not directly applicable to the problem of vulnerability analysis. Another reason this method is not applicable deals with the generation of the network. This network, which shows all of the dependent relations and probabilities, would have to be encoded with all of the possible vulnerabilities. As mentioned earlier, new technology is creating new vulnerabilities. If the Bayesian Network approach was used, it would require rebuilding at least a portion of the network structure each time a new vulnerability was identified or removed (Oliver90:387). Depending on the structure of the network, the changes may only have to be made to a single node. Changes in vulnerabilities cannot simply be spliced into or out of the existing network. There are several update methods that can be applied to making these changes, but if there are any intersecting nodes in the network, all of the update methods suffer from combinatorial problems (Oliver90:388). The existing network is built based on the dependencies of the known vulnerabilities. If changes are made to the network, they may have a propagation effect upon the existing dependencies with the net result being a complete rebuilding of the network.

2.3.4 Dempster-Shafer. Another methodology using probabilities is *Dempster-Shafer theory* (Dempster67, Shafer76, Giarratano89:275). Unlike Bayes' Theorem, where each event is treated individually, Dempster-Shafer works with sets of events. These sets of events are mutually exclusive and for each set of events a probability density function, m , is defined. In actuality, the probability density function is defined not only for the set of events, but for all subsets as well. If there are n events, there are 2^n subsets of events. According to Rich and Knight, many of these subsets of events are insignificant to the problem and their probability density functions return values of zero (Rich91:243).

The significance of this is that all of the combinations of vulnerabilities can be represented as sets of vulnerabilities. Some, if not most, of the subsets of vulnerability combinations would be impossible in the real world so the probability density function would return a value of zero.

Dempster-Shafer theory introduces the concepts of *belief* and *plausibility*. Belief is defined as the minimum support provided by the evidence while plausibility is the maximum support the evidence may be able to provide to the set of events (Giarratano89:284). Another concept introduced by Dempster-Shafer theory is *ignorance*. This concept allows for asserting a fact with a known confidence, but does not imply that the uncertainty of the fact is one minus the known confidence since there may be some confidence in the falsehood of an event occurring. In Dempster-Shafer theory, the probability density function can describe three aspects of a set of hypotheses. The first is the belief the set of hypotheses is true, the second that the set is not true, and the third probability distribution represents ignorance (Giarratano89:280).

A simple example of applying Dempster-Shafer would be to poll responses. If 100 people were polled on whether a law should be passed, some responses would be yes, some no, and some unknown or undecided. For discussion's sake, assume there were 40 yes responses, 35 no responses, and 25 undecided. Dempster-Shafer could make the following assertions. There is

belief that 40% of the respondents support passing of the law, but it is *plausible* that 65% of the respondents will support passing. There is also *belief* that 35% of the respondents support not passing the law, but it is *plausible* that 60% of the respondents will not support the passing of the law. There is also *ignorance* about 25% of the respondent's position. At some later point, the respondents with the unknown answers could commit to yes or no and then their *beliefs* could be attributed to the law passing or not passing respectively. Until such time though, there remains some *ignorance* about their responses.

A major problem with Dempster-Shafer is the generation of the probability density functions. Dempster-Shafer also requires a significant amount of *a priori* information (i.e., generation of the probability density functions), which if not supported with known probabilities, would have to be estimated.

2.3.5 Fuzzy Set Theory. The last approach discussed is *fuzzy set theory* (Zadeh65, Zadeh92). Fuzzy set theory, also known as fuzzy logic, is a generalization of normal set theory (Schmucker84:5). Normal set theory defines the membership of an element in a set as a Boolean predicate (i.e., yes or no). Fuzzy set theory represents the membership of a value in a given set as a possibility distribution (i.e., low to high) (Rich91:246). This variation allows definition of a set to represent an abstract concept such as tall.

In a normal set, a discrete element is defined for each height we wish to represent. This discreteness presents a problem if the question, "Is John tall?" is posed to the system. In normal set theory, it is very difficult, if not impossible, to adequately represent the concept tall. In fuzzy set theory, John has a membership value associated with the set TALL. John will also have a membership value associated with the sets MEDIUM and SHORT where TALL, MEDIUM, and SHORT all describe the height characteristics of people. TALL, MEDIUM, and SHORT are not necessarily disjoint sets and may overlap as shown in Figure 1. Depending on the definition of the

set characteristics, a person may have equal membership in multiple sets. For instance, using the sets defined in Figure 1, a person who is 5.75 feet tall would have the same membership value in the sets MEDIUM and TALL.

Continuing with the example of John's height, there is another difference between Boolean and fuzzy set theories. If John's height is five feet eleven inches, and in our Boolean system tall has been defined to be those persons six feet and over, again pose the question "Is John tall?" to the system. The Boolean system would reply no, although most human observers would tend to categorize John as tall. In the fuzzy logic system, John's membership in the set TALL is defined as something less than 1.0, but much greater than 0.0; probably around 0.99. The same question posed to a fuzzy logic system should reply that John is a member of the set TALL with a membership value of 0.99. John does not fully belong to the set TALL, but TALL would be a fairly accurate linguistic term to describe John's height.

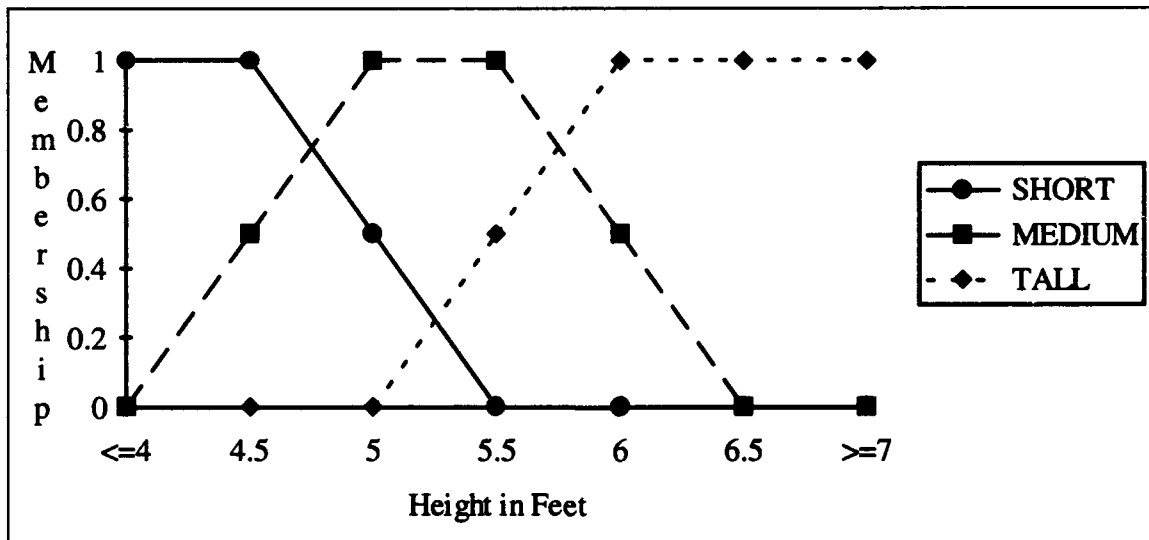


Figure 1. Fuzzy Sets Describing Height.

What these two examples demonstrate are the basic concepts exemplified by fuzzy set theory: the concepts of partial membership within a single set and membership in multiple sets describing the same attribute. What these concepts provide is a fairly easy and intuitive method to describe uncertain information.

Fuzzy set theory also allows modifiers such as *very*, *somewhat*, and *slightly* (Negoiu85:75). These modifiers can be defined to have the properties of concentrating, dilating, or shifting the primary fuzzy set definition. These modifiers allow for a more detailed discrimination of the members of a primary fuzzy set.

As pointed out earlier, fuzzy set theory is a generalization of normal set theory. As such, normal sets can be modeled using fuzzy set theory. This allows for the use of precise set definitions for those data items that are precisely defined and fuzzy set definitions for those data items that are ill-defined. An example would be the sets MALE and FEMALE. Most people would agree that these sets are precisely defined, genetic abnormalities aside, and as such constitute Boolean sets. In fuzzy set theory it is perfectly acceptable to pose the question, "Is John male and tall?".

Fuzzy set theory is not without its faults. Depending on how the set combination functions such as union and intersection are implemented, the internal representations of the results can suffer the same combinatorial explosion problems as the aforementioned methods. There are implementations, such as the table lookup method presented later in this thesis, which eliminate the combinatorial problems.

Fuzzy set theory, as applied to vulnerability analysis, still requires the *a priori* definition of the probability, or likelihood in fuzzy set terms, that the vulnerabilities will occur. The primary difference with assigning these likelihoods in fuzzy set theory is the use of linguistic terms. The other methods require that the probability values be discrete, numeric values, while in fuzzy set

theory, the likelihood indicates a *range* of values. This range, represented by a linguistic term, encompasses the inherent imprecision often found in security analysis (Schmucker84:20).

While all of these methods have similar capabilities and drawbacks, I found that fuzzy set theory seems to provide the most acceptable method for modeling and analyzing vulnerability data. Fuzzy set theory, at least in concept, can use natural language to quantify and reason about concepts describing ambiguous characteristics of an object or event (Giarratano89:291). The choice of fuzzy set theory to analyze vulnerabilities is not without support from other researchers. According to one researcher in the use of natural language for risk estimation, the increase in accuracy of the overall estimates by using natural language ranged from 16% to 32% (Nagy81). The use of natural language values helped to eliminate the extremely inaccurate estimates (Schmucker84:36).

2.4 Analysis Methods

2.4.1 Introduction. There are two general methods to analyze vulnerabilities: quantitative and qualitative. The quantitative method involves assigning numeric values to the attributes of the vulnerabilities and then using statistical and probabilistic techniques to evaluate the vulnerability of a system. The qualitative method involves assigning value judgments, also known as *linguistic values*, to the attributes of the vulnerabilities and then a technique such as fuzzy set theory is used to evaluate the vulnerability of a system.

As pointed out by Wood, *et.al.*, security experts differ in opinion as to which method (quantitative or qualitative) is the best for the evaluation of computer security (Wood87:7) and Schmucker indicates that there is no "established or standard" way to perform the evaluation of computer security (Schmucker:43). The reasons for this are threefold. First, many experts are

opposed to change. If an expert has been using a particular method throughout his career, he is not likely to want to learn, or possibly even recognize, another method. Second, there tends to be a significant cost difference between the two methods (Wood87:7). This cost difference can be attributed to the amount of information required for each method. Third, because of the extremely subjective nature of analyzing computer security vulnerabilities, no two experts can agree on the meaning attributed to results produced by any particular method.

The discussion below outlines the efforts of other authors to develop systems capable of dealing with this uncertainty in performing vulnerability and risk analysis. Most of these systems are risk analysis systems where vulnerability analysis, if present, is a sub-component of the overall system. There is no definitive method to perform the vulnerability analysis portion of a risk analysis.

Wood, *et.al.*, advocates the use of a weighted average of all vulnerabilities (Wood87:12). Schmucker also uses a weighted averaging scheme, but vulnerabilities are represented by a category and component hierarchy with the weighted average propagated up the hierarchy (Schmucker84:47). Hoffman and Neitzel follow an approach identical to Schmucker's (Hoffman80:366). The three systems above all use an estimate of the probability that the vulnerability will occur.

Wong advocates a system that uses past historical data to determine the frequency that a vulnerability occurs and modifies this frequency with a weighting factor to try to predict when the vulnerability might occur in the future (Wong77:98). For vulnerabilities with no historical data, Wong recommends using a statistical survey (Wong77:101). The last system identified is one proposed by Carroll. Carroll's system is business oriented and calculates the annual loss expectancy and return on investing in security measures (Carroll84:5).

2.4.2 Quantitative Analysis. Three of the identified risk analysis methods are based on quantitative analysis: Wong, Carroll, and Wood. Wong's method is based on the "loss unit concept" and is shown in Figure 2. Wong uses the term risk interchangeably with vulnerability. Once values have been derived for all of the frequencies of occurrence (F) and the range of consequential loss (L, L'), the sum of the products ($\sum F_i * L_i$) is calculated to produce the expected loss while the sum of the products ($\sum F_i * L_i'$) is calculated to produce the maximum expected loss. To predict future losses, a weighing factor is multiplied to each F_i , L_i , and L_i' . These weighing factors relate past frequencies of occurrence and consequential losses to predicted future values (Wong77:98). This method makes no attempt to analyze the vulnerabilities, what Wong calls risks, for importance or impact. The only measure used is the frequency of occurrence.

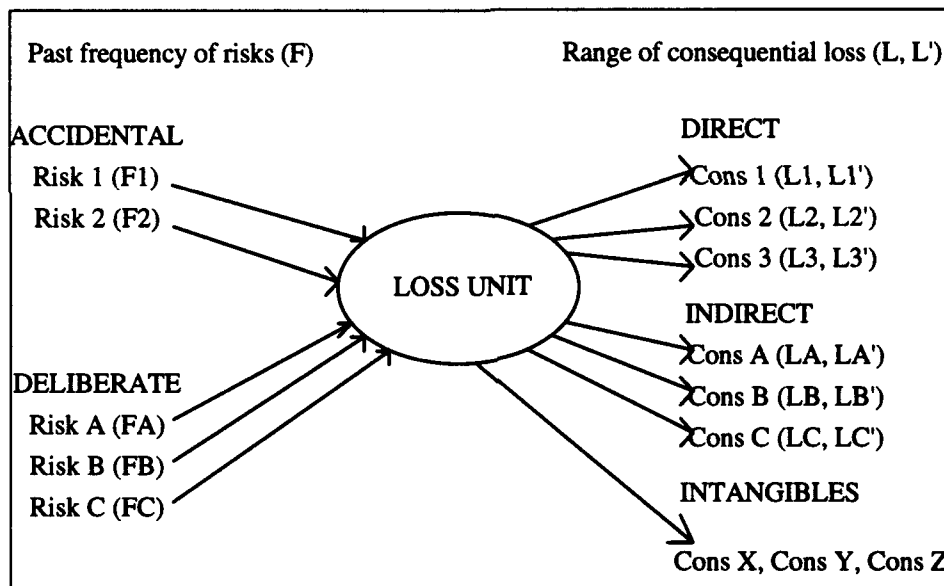


Figure 2. Loss Unit Concept (Wong77:94).

Carroll's risk analysis method is based on calculating the return on investment given by placing security measures in place. This return on investment is calculated by computing the difference in annual loss expectancy without security measure and the annual loss expectancy with

security measures. The annual cost of security measures is subtracted from this difference to provide the return on investment (Carroll84:5). One of the values used by Carroll to calculate the annual loss expectancy is a vulnerability rating. This rating is calculated by assigning a *subjective* value from 1 to 5 representing the severity of a vulnerability. This value is then converted using a logarithmic scale to a value between 0 and 1. Carroll contends, but provides no supporting references, that "human judgment tends to be quite accurate at the lower end of a subjective scale but not so good at the upper end." (Carroll84:91).

The last quantitative risk analysis method presented is Wood, *et.al.* Wood, *et.al.*, have identified the most prevalent vulnerabilities present in most computer systems and provide these vulnerabilities in a check-list format. In this method, each vulnerability present in a system is scored with a numeric value of 0.9, 0.7, 0.5, 0.3, or 0.1. These values represent the linguistic equivalent of Very High, High, Medium, Low, and Very Low respectively. Once the check-list has been completed, the ratio of vulnerabilities present to the vulnerabilities listed in the check-list is calculated for each possible score. A weighted fraction is then calculated, as shown in Equation (2), by summing the products of the ratios with their respective scores and dividing by the sum of the scores.

$$FW = \frac{\frac{NR(0.9)}{N(0.9)} \times 0.9 + \frac{NR(0.7)}{N(0.7)} \times 0.7 + \dots + \frac{NR(0.1)}{N(0.1)} \times 0.1}{0.9 + 0.7 + 0.5 + 0.3 + 0.1} \quad (2)$$

where: NR(x) = number of relevant vulnerabilities for score x
N(x) = number of vulnerabilities in check-list for score x (Wood87:12)

This reciprocal of this weighted fraction is calculated and is referred to by Wood as the applicability index. A non-adjusted score is calculated by taking the sum of the products of the number of relevant vulnerabilities and their respective scores (i.e., NR(0.9)*0.9 + NR(0.7)*0.7 +

... + NR(0.1)*0.1). A maximum possible score is calculated by taking the sum of the products of the number vulnerabilities in the check-list for each possible score and their respective scores (i.e., $N(0.9)*0.9 + N(0.7)*0.7 + \dots + N(0.1)*0.1$). The adjusted score is computed as the product of the non-adjusted score and the applicability index. The final calculation produces a what Wood calls a control comprehensiveness indicator and is the ratio of the adjusted score and the maximum possible score. This values is to be used as an indicator of how well the system security measures perform with respect to the vulnerabilities identified in the checklist (Wood87:13-16).

As an overview of the quantitative methods presented, a few of the benefits and detractors of these methods should be mentioned. First, significant effort must be put into acquiring the numerical probability that each vulnerability will occur. Also if any of the vulnerabilities are dependent on each other, the dependent probabilities must be determined. Obviously, this requires an extensive statistical database that covers all of the vulnerability combinations (Wong77:83). The quantitative methods also suffer when this statistical information is not available for a given vulnerability. In fact, when no statistical information is available, the assignment of the probabilities of occurrence becomes subjective(Wong77:101). In other words, the expert, in the absence of statistical data, makes a qualitative value assignment and translates that value into a numeric value. If a comprehensive statistical database is available, the quantitative methods tend to remove the subjective biases induced through estimation (Wong77:84).

2.4.3 Qualitative Analysis. In qualitative methods, value judgments are associated with each vulnerability. These value judgments are often based on the instinct, intuition, and experience of the expert performing the evaluation, but may include some statistical bias. For instance, if an expert knows from experience that users often use common words for passwords, then the vulnerability caused by not changing passwords frequently may have a higher importance than users writing down their passwords. If on the other hand, the expert knows that the system

generates random passwords, the vulnerability of users writing down their passwords would be higher than that of infrequent password changes.

The two remaining risk analysis systems identified are based on qualitative analysis methods. These systems are identical in their approach in that they use fuzzy logic and linguistic variables to perform the analysis. Both the Schmucker system and the Hoffman and Neitzel system use a fuzzy weighted average. Each vulnerability is assigned three linguistic values to represent possibility of loss, severity of loss, and reliability of estimate. The vulnerabilities are represented in a hierarchical network to indicate the interdependence of vulnerabilities and also to indicate the various sub-components of the system being analyzed. For each parent node in the hierarchy, the weighted average of the children nodes is calculated and assigned as the risk of the parent node. This process is repeated with the weighted averages propagated up the hierarchy until a single risk value is generated for the top node (Schmucker84:45-47, Hoffman80:370). Schmucker goes into great detail as to how to perform this weighted average (Schmucker84:49-55) while Hoffman and Neitzel simply provide a conceptual overview (Hoffman80:370).

2.5 Organization of Vulnerability Information

Regardless of whether quantitative or qualitative methods are used, there are several formats for organizing the vulnerability information. Wong advocates the use of a top-down hierarchy to represent the entire structure of vulnerabilities (Wong77:7). This top-down structure is built using information from a variety of sources that include procedures, company and account information, contract information, site inspections, interviews, and functional flowcharts (Wong77:43). Combining all of these sources, the analyst is able to generate a hierarchy of vulnerabilities. Schmucker also advocates the use of a top-down hierarchy to represent vulnerabilities. This

hierarchy is based on the decomposition of the computer system into components and dependent vulnerabilities (Schmucker84:45).

Another method used to represent vulnerabilities is a check-list. ARES is an example of such a check-list system. In this type of system, all of the possible vulnerabilities are known and the analyst merely indicates the presence or absence of each vulnerability. ARES suffers in that the importance of each vulnerability is not known and that no real vulnerability analysis is performed. Wood, *et.al.*, also provide a check-list approach. Wood, *et.al.*, generated their list of vulnerabilities based on work performed for the Lawrence Livermore National Laboratory and the former USAF Logistics Command. This checklist contains 857 identified vulnerabilities (Wood87:14) where a predetermined importance value is associated with each vulnerability (Wood87:10). Carroll also provides a check-list though not as comprehensive as Wood, *et.al.*

2.6 Current Air Force Analysis Tools

As mentioned earlier, ARES was an attempt to close the gap in computer security between the number of risk analyses the security managers could perform and the number of systems requiring analysis. However, as with any computer product, users identified several shortcomings. Foremost, ARES provided no method to evaluate identified vulnerabilities. ARES simply provided a list of vulnerabilities to the CSO partitioned into several categories: *Audit Trails, Backup, Contingency Plan, Documentation, Information Access Control, Magnetic Remanence* (i.e., traces), *Media Storage and Control, Operating System, Passwords, Physical Security, Small Computers, Software Configuration Management, and Security Test and Evaluation.*

Within a category, ARES did not differentiate between vulnerabilities based on their potential impact to the system security. For example, ARES did not differentiate between a door

with no lock in an unclassified environment and a system processing Top Secret data with no passwords. Both were listed as vulnerabilities under Access Control. It was up to the CSO to determine which vulnerability was more important. Because of this failing, a CSO may overlook serious flaws in a system if many minor flaws are mixed in.

To aid in fixing this shortcoming, AFTT proposed a long term multi-phase project. The first phase of the project identifies, evaluates, and develops methods to evaluate computer security vulnerabilities. Other phases will deal with implementing and integrating the methods into ARES or a similar program and expanding the use of automated methods into performing the valuation and risk assessment of the overall system.

2.7 Automating Computer Security Analysis

There are three basic problems hampering the automation of computer security analysis: secrecy, attitudes about security, and changing technology. Computer security analysis is often shrouded in secrecy. The reasons for this secrecy are threefold. First, no computer facility is absolutely secure. Consequently, no security analysts will reveal what vulnerabilities exist at their site. Second, when a security analyst develops a method to determine the level of security at his site, he often will not publish the results. This is to prevent the security analyst's opponents from using the method to determine his security vulnerabilities. Last, security analysis, at least in the past, can be considered a "black art". A security analyst often makes decisions about the relative security of a site based on instinct, intuition, and experience rather than rules and formulas. Given this subjective analysis, it is very difficult to automate security analysis.

Another reason computer security analysis has not been successfully automated is the attitude about security in general. In the research community, computer security is often viewed as

a hindrance. Users often want to share their findings with many colleagues and thus will try to circumvent safeguards to allow freer access to their data and system. Users also tend to have very myopic views about the scope of security. They may intend to only allow access to their files by selected colleagues when in fact they open up the entire system to everyone. At most sites, the only personnel who are truly security conscious are the system administrators and CSO.

The last hurdle in trying to automate computer security is changing technology. Twenty years ago, most computers were large mainframes with many terminals connected. Outside access to the systems was minimal if existent. Today, computers are connected worldwide via high speed networks. With the speed of these networks reaching 100 megabytes per second transfer rate, it only takes a few seconds for large amounts of data to be compromised.

The other change in technology is the proliferation of personal computers. As of 1988, over 45 million personal computers were in use. In that same year, personal computers made up 92% of the total computers shipped by US manufacturers. For the five year span of 1987 through 1991, US manufactures shipped almost 32 million computer systems; personal computers accounting for 29.5 million of the systems. The government alone had a ten fold increase in the use of personal computers. Also of interest was the increase in the number of workstations sold during this 5 year period: while mainframe sales decreased by 62%, workstation sales increased by 360%. (Census93:Tables 648, 1273, and 1274)

What this implies is that methodologies developed to ensure the security of large mainframes with dedicated terminals may not necessarily work for large distributed networks with many individual personal computers and workstations.

Given these problems, the need for automating computer security has never been greater. Computer security analysts need effective and reliable tools to help them evaluate the risks and

vulnerabilities associated with computers. They need tools that are adaptable to new technologies and scalable to handle the increasing number and types of systems requiring evaluation.

2.8 Summary

There is a definite need to perform timely and accurate evaluation of computer security vulnerabilities. The ability to handle uncertain information is paramount in performing this evaluation. The evaluation method should not require an extensive statistical database in order to establish a baseline for processing. Because of increasing numbers of computer systems and thus the required number of evaluations to be performed, the evaluation process needs to be automated.

Much emphasis is placed on performing risk analysis of computer systems, but little emphasis is placed on vulnerability analysis. As one author stated, "The process of risk analysis centers on vulnerabilities" (Carroll84:87).

Of the tools available to assist in automating the evaluation of computer security vulnerabilities, those AI tools with the capacity to reason with uncertainty appear to hold promise. There are many AI methods that provide the capability to evaluate uncertain information, a few of which have been presented in this chapter. Of these, I find fuzzy set theory the most interesting approach and it appears to have promise for dealing with the inherent imprecision found in security analysis. Fuzzy set theory has the capability to describe and maintain the relationship between two facts, whether well- or ill-defined, and through the use of linguistic variables, provides an intuitive (to the author) language-based interface between the system and the security analyst.

III Methodology

3.1 Introduction

The methods used to evaluate computer security vulnerabilities can be divided into two main types: quantitative and qualitative. Quantitative methods involve assigning a numerical value to each criterion under consideration and usually involve probabilistic or statistical analysis. Qualitative methods assign a linguistic value to each criterion. These linguistic values include terms such as high, low, likely, possible, and never. A qualitative analysis method that uses linguistic terms is fuzzy logic. With either approach, the assignment of a value, whether a numeric quantity or linguistic quality, is subjective in the absence of an extensive statistical database.

3.2 Overview

This chapter outlines the methodology used to develop an AI method to assist in the evaluation of computer security vulnerabilities. It is my contention that a qualitative approach will provide comparable results: it is computationally feasible, theoretically sound, scalable, easier to use, and more intuitive to the user. The main result of this research is the demonstration of the feasibility of using a qualitative analysis method to evaluate computer security vulnerabilities.

In order to demonstrate the feasibility of using a qualitative analysis method to evaluate vulnerabilities, a comparable quantitative analysis method had to be identified or developed. As discussed in chapter 2, the available research focused on risk analysis methods, not vulnerability analysis. Where vulnerability analysis was mentioned, it involved the subjective assignment of values to the vulnerabilities. Not finding any existing vulnerability analysis system, I developed

my own. Without any industry standard to benchmark my development against, the methods I developed and present here are based on the intuition and experience I developed as a computer systems analyst for and manager of a very large, secure data processing facility.

To ensure the vulnerability analysis system I developed wasn't biased towards my hypothesis, I developed the quantitative analysis method first. This system is based on standard statistical methods of analysis for independent vulnerabilities. Independence of vulnerabilities is assumed to minimize the effects of combinatorial explosion. Although in the real world, vulnerabilities are not necessarily independent, I felt it was best to keep the systems simple. This simplicity makes the results easier to compare.

I then implemented a qualitative method using the same assumption of independence, but using fuzzy set theory as the primary analysis method. I chose fuzzy set theory because I felt it best represented a fully qualitative approach. I based my initial fuzzy set theory implementation on the work of Schmucker, but discovered that this method suffers from combinatorial explosion. To overcome this, I developed an alternative table lookup method. The reasons and justification for using this table lookup method are provided later in this chapter.

One concern was the applicability of this research to existing risk analysis systems. As most of the existing systems identified simply require a subjective vulnerability value, I contend that a systematic analysis of identified vulnerabilities should increase the accuracy of these risk analysis systems. This contention is made based on intuition as opposed to empirical evidence. Due to time and budgetary constraints, I was not able to obtain a working implementation of any of the mentioned risk analysis systems in order to prove my contentions. However, I feel that any systematic approach that is reasonable is better than a subjective guess.

3.3 Background

There are two main methods I used in this thesis to evaluate vulnerability data. The first is a quantitative method based on statistical analysis. The second is a qualitative method based on fuzzy logic as proposed by Zadeh (Zadeh65:338). Below are the specifications of the methods used.

Each method implemented used a subset of the vulnerabilities identified by trial runs of ARES. This data, shown in Appendix B, was only used as a demonstration vehicle for each method and is the result of many varied runs of ARES. The probability distributions and influence values associated with the identified vulnerabilities were randomly generated and, therefore, the vulnerability values shown do not represent an actual system. However, the data is representative of a possible system.

3.4 Statistical Analysis.

The statistical analysis method used is straight-forward and should be familiar to most readers. The main text used for this analysis method is "Probability and Statistics for Engineers" by Scheaffer and McClave (Scheaffer86:1). For each vulnerability, eight values are given. The first value, 'vuln-influence', given with each vulnerability indicates the overall influence of that particular vulnerability across all seven functional areas. This value, between 0 and 1, represents the subjective rating of importance of this vulnerability with respect to the overall vulnerability of the system. The rating value given assumes that the vulnerability is independent from all other vulnerabilities. The other seven values, indicated by the prefix 'dist-', represent the allotment of

this vulnerability's influence to each of the defined functional areas presented in the previous chapter. Each of these allotment values range from 0 to 1 and indicate how much of this vulnerability's influence value is applied to a specific functional area. These allotment values can also be thought of as an impact rating indicating the degree to which this vulnerability impacts a particular functional area.

Two separate processes are performed on the data. The first is a statistical analysis for each functional area. For each functional area, the total influence given, the weighted average of influence given, the standard deviation of influence, and the percentage of influence are calculated. The product of the influence value and the functional area distribution value is used as the contribution by each vulnerability. The contributions of all vulnerabilities to a specific functional area are summed to generate the total influence given to that functional area. This sum is then divided by the number of vulnerabilities contributing to produce the weighted average of influence given. The standard deviation of influence is calculated using the contributions of each vulnerability for a specific functional area. The percentage of influence is calculated by taking the total influence given for a functional area and dividing by the sum of the total influence given for all areas.

The second process performed on the data involves identifying those vulnerabilities that contribute significantly to each functional area. This is done by identifying and listing those vulnerabilities that are in or matching the top 10% of the contributors (i.e., if 50 vulnerabilities are in the system, then at least the top 5 contributors, but maybe more depending on tying conditions). Tying conditions prevent the strict application of a 10% cutoff. With no other information available other than contribution, it is not reasonable to discriminate against a vulnerability if its contribution value ties with one in the top 10%.

Also identified are those vulnerabilities that are in or matching the top 10% of more than one functional area. It should be apparent that the vulnerabilities that occur in or match the top 10% of the most functional areas are the most critical. The purpose of this is to identify to the computer security expert those vulnerabilities that should be addressed first.

Groupings of vulnerabilities is also performed based on units of standard deviation as measured from the maximum contributor. In other words, those vulnerabilities within 1 standard deviation, 2 standard deviations, and so forth. The use of the standard deviation is an arbitrary choice, but does provide some sense as to how the vulnerabilities could be grouped.

3.5 Fuzzy Logic.

For the fuzzy logic analysis, two approaches were implemented. The first was a calculation method based on the method implemented by Schmucker and the second approximated these calculations using a table lookup.

In both methods, it is impossible to take a weighted average of linguistic values since the definition of weighted average implies division by the number of terms in the summation. For fuzzy logic, it is more appropriate to use a normalized average. In the statistical analysis method, the sum of the contributions to a functional area was divided by the number of vulnerabilities contributing to that functional area to produce a weighted average of influence. This was possible since the maximum contribution of any one vulnerability to any functional area was one and the sum of these maximum contribution equals the number of vulnerabilities. In the fuzzy logic analysis, this weighted average is simulated by normalizing the sum of contributions by the sum of the influence values for each vulnerability. The sum of influence values in the fuzzy logic analysis method is equivalent to the sum of the maximum contributions in the statistical analysis method.

As in the statistical analysis method, two separate processes are performed on the data. The first provides the total influence given to the entire system, the influence given to each functional area, and the normalized average influence given each functional area. Since linguistic values are used, it is not possible to generate a standard deviation or percent of influence given.

The contribution of each vulnerability to a functional area is again the product of the vulnerability's influence value and distribution value. The sum of these contributions make up the total influence given to each functional area. The sum of the total influence given to each functional area gives the total influence given the system. The normalized average influence is calculated as the total influence given each area divided by the total influence given the system.

The second process performed on the data is again the identification of those vulnerabilities that provide significant contribution to each functional area. The same top 10% criteria is applied to the fuzzy results as was applied to the statistical results in order to identify critical vulnerabilities. Since the standard deviation cannot be calculated, the linguistic values themselves are used to group the results. What follows is a discussion of the two methods used to perform the fuzzy arithmetic and a discussion of fuzzy arithmetic in general.

3.5.1 Schmucker's Calculation Method. In the first approach implemented with fuzzy logic, the calculations were based on Schmucker (Schmucker84:48) and used the equations shown in Equation (3) for fuzzy arithmetic. In the equations the notation $a(i)/i$ represents the fuzzy element i in the set with membership in the set of $a(i)$. Schmucker explains his fuzzy arithmetic equations as follows:

What this definition means computationally is that to compute the degree of membership of, say, 8 in $A+B$, we have to examine all of the possible ways that two integers (taken from the set $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$) can sum to 8 and examine the degrees of membership of these pairs. Thus, if the degree of membership of 8 in $A + B$ was x , then x would be computed as follows:

$$x = \max\{\min(a(1), b(7)), \min(a(2), b(6)), \min(a(3), b(5)), \min(a(4), b(4)), \min(a(5), b(3)), \min(a(6), b(2)), \min(a(7), b(1))\}.$$

Each of the minimum operations computes one of the degrees of membership of 8 in the set $A + B$. We then take the greatest such degree of membership to be *the* degree of membership of 8 (Schmucker84:48).

$$\begin{aligned} A &= \{a(i)/i \mid 1 \leq i \leq n\} \\ B &= \{b(j)/j \mid 1 \leq j \leq n\} \\ A+B &= \max\{\{\min(a(i), b(j))\} / [k] \mid 1 \leq i, j \leq n, k = i + j\} \\ A * B &= \max\{\{\min(a(i), b(j))\} / [k] \mid 1 \leq i, j \leq n, k = i * j\} \\ A / B &= \max\{\{\min(a(i), b(j))\} / [k] \mid 1 \leq i, j \leq n, k = i / j\} \end{aligned} \quad (3)$$

where: i, j , and k are fuzzy set indices
 n is the number of elements used to describe the fuzzy set

Here, the fuzzy set is defined over n elements. Although fuzzy sets can be defined over continuous functions, it is much easier to implement using a discretized set. The foundation for these equations is based on Zadeh's extension principle and a good explanation is given in Appendix B of Schmucker's book (Schmucker84:133). Similar methods for fuzzy arithmetic are outlined by Kaufmann and Gupta (Kaufmann85:14)

Before continuing, an example of how these equations operate is beneficial. For simplicity's sake, we will define three sets with three discrete points.

$$\begin{aligned} \text{ONE} &= \{1/1, 0/2, 0/3\} \\ \text{TWO} &= \{0/1, 1/2, 0/3\} \\ \text{THREE} &= \{0/1, 0/2, 1/3\} \end{aligned}$$

If we add ONE and TWO we get the following:

$$\text{ONE} + \text{TWO} = \{0/2, 1/3, 0/4, 0/3, 0/4, 0/5, 0/4, 0/5, 0/6\}$$

and multiplication results in:

$$\text{ONE} * \text{TWO} = \{ 0/1, 1/2, 0/3, 0/2, 0/4, 0/6, 0/3, 0/6, 0/9 \}$$

It should be noted that missing elements are assumed to have a membership of zero and for terms with multiple indices, the maximum membership value is used. Therefore, the resulting sets, after correcting the terms are as follows:

$$\text{ONE} + \text{TWO} = \{0/1, 0/2, 1/3, 0/4, 0/5, 0/6\}$$

$$\text{ONE} * \text{TWO} = \{0/1, 1/2, 0/3, 0/4, 0/5, 0/6, 0/7, 0/8, 0/9\}$$

Based on the equation above, the result of a division is the following.

$$\text{TWO} / \text{ONE} = \{0/1, 0/0.5, 0/0.33, 1/2, 0/1, 0/0.67, 0/3, 0/1.5, 0/1\}$$

Division is used by Schmucker to produce a weighted "average" of the influence in his vulnerability analysis. To simplify the results of fuzzy set division, Schmucker uses a method proposed by Clements. This method places in the set resulting from the fuzzy division, only those terms i and j , which when i is divided by j result in an integer. The example below demonstrates this simplification.

$$\text{TWO} / \text{ONE} = \{0/1, 1/2, 0/1, 0/3, 0/1\}$$

As before, should multiple fuzzy set indices occur, the index with the maximum membership is used in the final normalization of results. This results in the following:

$$\text{TWO} / \text{ONE} = \{0/1, 1/2, 0/3\}$$

The major drawback of implementing Schmucker's method is how the fuzzy sets expand. For instance, if 50 fuzzy sets, each defined over 7 terms, are added, the resulting set is defined over 350 terms. The expansion problem is greatly exacerbated for multiplication. If the same 50 sets are multiplied together, the resulting set contains 7^{50} or almost 1.8×10^{42} terms.

This is clearly not feasible for vulnerability analysis where the possibility exists for many hundreds of vulnerabilities. Even using the weighted averaging function defined by Schmucker,

which will return a final result fuzzy set with the same number of terms as the primary sets (Schmucker84:49), the intermediate calculations within the averaging process tend to make the fuzzy set calculations intractable.

An example demonstrates this very quickly. Assume 50 identified vulnerabilities, each vulnerability distributed over 7 functional areas and containing a single vulnerability influence value. To calculate the weighted average for a single functional area would require that the numerator of the weighted average function contain a fuzzy set definition with 2450 elements and denominator contain 350. This assumes that the primary fuzzy set is defined over 7 elements. The numerator consists of adding 50 49-element sets. The 49-element sets are generated by multiplying 2 7-element primary terms (the distribution value for a functional area and the influence provided by that vulnerability). The denominator is the sum of 50 7-element sets.

For each iteration through the equation, i.e., for each i and j , two operations are required. The first is the mathematical operation on the index and the second is the comparison for the minimum value at $a(i)$ and $b(j)$. Each multiplication of the 2 7-element sets requires 98 operations. The first addition of 2 49-element sets requires 4802 operations (49 indices in first set, 49 indices in second set, 2 operations per index) resulting in a 98-element set after normalization. For the sake of simplicity, we will assume normalization requires zero operations. The second addition requires 9604 operations; the third requires 19208 operations and so forth. For this example, the number of operations required for each successive addition grows on the order of 2^{n-1} . Clearly, this is not computationally feasible for any large number of vulnerabilities.

3.5.1.1 Translation of Fuzzy Set to Linguistic Term. After a set has been normalized, it is 'translated' to a linguistic term. In this case, a "best fit" approach is used to translate a fuzzy set to a linguistic term. The equation to do this best fit is given in Schmucker (Schmucker84:56) and is shown in Equation (4).

The actual translation occurs when the Euclidean distance is calculated for each of the pre-defined fuzzy terms. The fuzzy term with the smallest distance from the set of interest is considered the "best fit".

$$d(X, F) = \sqrt{\sum_{i=1}^n (x(i) - f(i))^2} \quad (4)$$

where: **X** is fuzzy set to be translated

F is fuzzy set representing a pre-defined linguistic term

There is an ambiguity with this method. The problem occurs when the Euclidean distance is the same for two or more fuzzy sets. As implemented, the system will choose the linguistic term with the 'lowest' relative value. This in part is to prevent the system from suffering combinatorial explosion.

3.5.1.2 Normalization and Convexity of Fuzzy Sets. This implementation made use of normalized and convex sets. The use of normalized and convex fuzzy sets aids in 'translating' a fuzzy set back to a linguistic term as shown above.

A *normal fuzzy set* is where the element(s) of the set with the maximum membership value has (have) a membership value of one. A fuzzy set *A* is *normal* if and only if

$$\forall x \in R; \max_x \mu_A(x) = 1$$

where $\mu_A(x)$ represents the membership value of element *x* in fuzzy set *A* (Kaufmann85:12).

A *convex fuzzy set* is where if the set was plotted, it would have at most one positive slope and at most one negative slope. Note that the plot of the set is not required to have either (e.g., a horizontal line), or may have only a single positive or a single negative sloping line, or may have both, but it cannot have more than one positive or more than one negative sloping line if the fuzzy set is to be convex.

A fuzzy set A is *convex* if and only if,

$$\forall x, y \in R: \mu_A[\lambda x + (1 - \lambda)y] \geq \mu_A(x) \wedge \mu_A(y), \forall \lambda \in [0, 1].$$

where $\mu_A(x)$ and $\mu_A(y)$ represent the membership values of elements x and y respectively in fuzzy set A (Kaufmann85:11). The purpose of making a set convex is to prevent conditions like High and Low from occurring simultaneously. Of course, this could be represented by Not Medium, but that is an implementation choice.

I did however, make use of convex sets in my implementation of Schmucker's fuzzy arithmetic. This was particularly important for the fuzzy multiplication of two seven-element fuzzy sets that returned a set of 49 terms. These sets usually had many peaks and valleys. Depending on where these peaks and valleys fell, the translation back to a linguistic term would produce unreliable results. For instance, Very High * Very High would return Very Low. This is because once the set was mapped back to a 7-element set, the value at element 1 would be greater than any other value. By making the 49-element set convex before mapping back to a 7-element set, this eliminated that problem.

There is a concern that normalizing a fuzzy set and making a fuzzy set convex will change the meaning of the fuzzy set prior to these operations being performed. The reason these operations are used, even though they may change the meaning of the original fuzzy set, is because without ensuring the convexity and normalization of the fuzzy set it is extremely difficult to find a linguistic expression to represent the original fuzzy set. As pointed out above, the use of convexity and normalization prevent the case of a fuzzy set being described as both High and Low but not Medium.

3.5.2 Table Lookup Method. The second approach implemented using fuzzy logic attempted to overcome the combinatorial explosion. To do this, a table lookup method was

implemented to perform the fuzzy mathematical operations. The values for the addition and multiplication fuzzy function tables are shown in Tables 1 and 2 starting on page 51. These tables were derived using a method that attempted to model the behavior of the equivalent numeric functions. The behavior of the function is such that adding a small (relative magnitude between 0 and 1) number to a small number produces a small number. Likewise, adding a large number to a large number produces a large number.

The translation of the numeric value into its equivalent linguistic term was accomplished using both a linear mapping and a non-linear mapping. The linear mapping was produced by dividing the range of 0 to 1 into even groups. In this case, I had seven linguistic terms, so each group equaled one-seventh. Any value between 0 and $1/7$ was assigned to VERYLOW, any value between $1/7$ and $2/7$ was assigned to LOW, and so forth.

The non-linear mapping was based on the fuzzy distributions defined in Figure 3. These fuzzy sets were arbitrarily chosen with the intent to build a model with a large middle and small extremes. The goal of building this model was simply to test the effectiveness of a table lookup scheme using a non-uniform distribution mapping. Here, the assignment occurs to those linguistic values where the numeric value has a maximum membership. For instance, 0.03 would map to VERYLOW, but 0.04 would map to LOW.

To actually build the table, a simple program was built with two loops that iterated from 0 to 1 in increments of 0.001. The value of each loop was translated to a linguistic term in order to determine the fuzzy set these values were in. The indicated operation was performed on the numeric values and the result was translated to a linguistic term. By keeping track of how many of each linguistic result occurred given the linguistic input values, and using the linguistic result with the maximum occurrences, the function behavior was mapped.

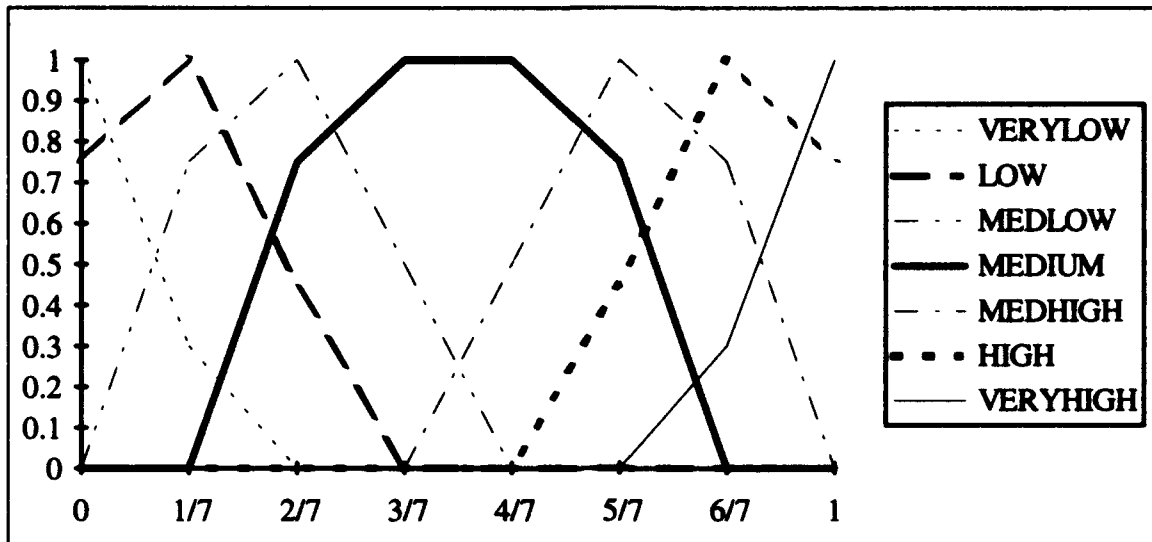


Figure 3. Defined Fuzzy Sets.

For instance, in building the linear mapping multiplication table, the input values might be 0.5 and 0.2. The 0.5 would map to MEDIUM and the 0.2 to LOW. The product of 0.5 and 0.2, 0.1 maps to VERYLOW. However, if the inputs were 0.56 and 0.28, again mapping to MEDIUM and LOW respectively, the result, 0.1568 maps to LOW. This is an example of the boundary condition caused by mapping an infinite sequence of values onto a finite map. As it turns out in this example, with the loops providing a 1000 values from 0 to 1, there are 19,138 VERYLOW products of MEDIUM and LOW and 1,311 LOW products. Because of the preponderance of VERYLOW results, that value is used to represent the result of multiplying MEDIUM and LOW. It should be pointed out that this method maintains the normal commutativity of multiplication and addition.

After repeated trials, it was determined that the linear mapping best represents the desired behavior of the addition and multiplication functions.

3.5.3 Fuzzy Arithmetic. Fuzzy arithmetic is created by using the extension principle outlined by Zadeh and discussed by Schmucker (Schmucker84:133). The extension principle

allows for any function to be mapped to fuzzy sets. By using the functional definitions of addition, multiplication, and division for real numbers and mapping these functions into the fuzzy set domain, the equations shown in Equation (3) on page 33 are derived.

The biggest problem in trying to implement an algebra defining fuzzy arithmetic involves the lack of an infinite domain space. The extension principle assumes the existence of an infinite domain space made up of all the possible fuzzy sets. This is a valid assumption for the theoretical generation of fuzzy functions, but leads to a problem in an actual implementation of a fuzzy algebra.

To demonstrate this, assume there exists three primary fuzzy sets: LOW, MEDIUM, and HIGH. Also assume a hedge or modifier has been defined: VERY. The hedge is an operation performed on a primary fuzzy set. The problem arises in the semantic meaning of the results when a hedge is applied repeatedly. Assume VERY is applied repeatedly giving the result of VERY-VERY-VERY-VERY-VERY-LOW, that will be represented here as $\text{VERY}^6 \text{LOW}$. While most would agree that there is a different semantic meaning to VERY LOW and $\text{VERY}^2 \text{LOW}$, there is little semantic difference between $\text{VERY}^6 \text{LOW}$ and $\text{VERY}^7 \text{LOW}$. What about $\text{VERY}^{100} \text{LOW}$ and $\text{VERY}^{101} \text{LOW}$? What is the difference between $\text{VERY}^\infty \text{LOW}$ and $\text{VERY}^{\infty-1} \text{LOW}$?

For the last question, I conclude there is none based on the definition that $\infty-1 = \infty$. Where then is the line drawn to represent difference? The line is drawn subjectively by the implementer of the fuzzy algebra, much as a programmer decides the number of significant digits used to represent real values. Given a subjective cutoff for significance, this defines a finite number of fuzzy sets that can be used to represent all the values possible in the implementation of the fuzzy algebra.

Another problem in implementing fuzzy arithmetic involves the semantic meaning of the operations. There is no single meaning that can be applied to performing an arithmetic operation

on fuzzy sets. Although there is an intuitive meaning to the term "addition", especially with real numbers, the intuition falls short for fuzzy sets. An example best demonstrates this lack of intuition.

Assume the problem requires adding HIGH and LOW. If it is assumed that both values are positively increasing then adding LOW to HIGH will only increase HIGH, possibly to VERY HIGH (see Figure 4). Another way to state this assumption is that only positive values can be represented. A real world example is vulnerability analysis. If the influence of one vulnerability is HIGH, and the other is LOW, then the total influence of both would be HIGH to VERY HIGH.

If on the other hand, it is assumed that one of the fuzzy sets represents a median value, then "adding" LOW to HIGH will result in a value of MEDIUM. This assumption has the effect of causing those fuzzy sets below (linguistically represent smaller values) the median value to be 'negative' from an additive point of view and those above (linguistically represent larger values) the median to be positive (see Figure 5). The net effect of adding two fuzzy sets under this assumption is to generate an 'average' of the values that the two fuzzy sets represent. Using this method requires an odd number of fuzzy sets be defined. If the influence of one vulnerability is HIGH and the other is LOW, then the influence distributed across both vulnerabilities is MEDIUM.

Both assumptions concerning the meaning of "adding" two fuzzy sets are valid, but mutually exclusive. The meaning implied by the "addition" of two fuzzy sets is subjective and implementation dependent.

The same ambiguities carry over into multiplication. Although the concept of multiplication when dealing with real numbers is simple, fuzzy set multiplication is not so easy to understand. Multiplication is nothing more than repetitive addition. The multiplication of the integer values 4 and 5 together can be stated as the addition of 4 to itself 5 times. The translation to fuzzy multiplication becomes confusing when HIGH is "added" to itself LOW times.

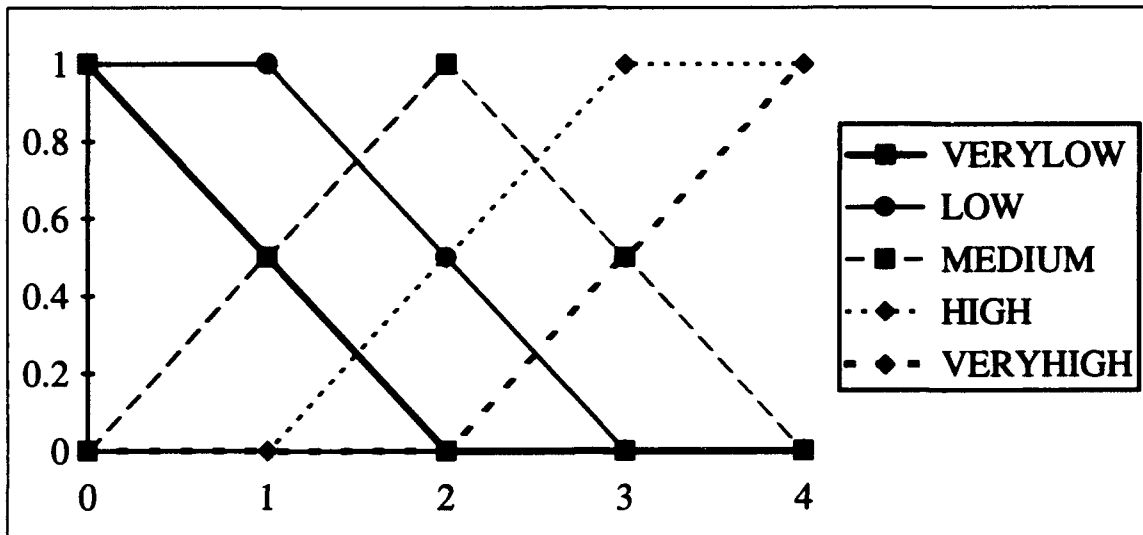


Figure 4. Positive Increasing Fuzzy Sets.

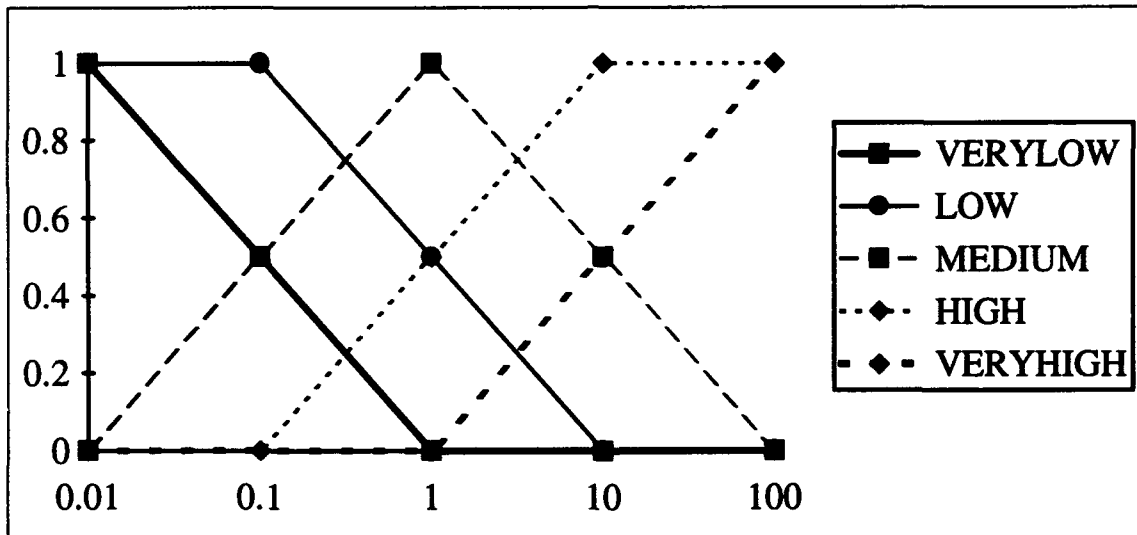


Figure 5. Median Based Fuzzy Sets.

If the first assumption concerning addition is used, then the multiplication of HIGH and LOW will result in a value of at least HIGH. If the second assumption is used, then the result will be somewhere near MEDIUM. Again these assumptions seem valid, but are mutually exclusive.

A third possibility is that multiplication performs a normalized weighting function. In other words, the result of multiplying two fuzzy sets together tends to 'shift' one of the sets towards the other. For instance, HIGH multiplied by LOW would 'shift' HIGH towards LOW and result in a value of MEDIUM HIGH while LOW multiplied by HIGH would 'shift' LOW towards HIGH and result in a value of MEDIUM LOW. Immediately, the reader should notice that depending on how much "shift" is caused by the operation, the commutative law may not hold. If commutativity is important to the implementation, the "shift" could be symmetric. Also, in this example, the first term of the multiplication is 'shifted' by the second.

The best way to see how this works is with an example. Suppose we are trying to determine the magnitude of a budget. If the HIGH cost items only occur a LOW number of times, then the contribution of the HIGH cost items is MEDIUM HIGH. Conversely, if the LOW cost items occur a HIGH number of times, the contribution of the LOW cost items is MEDIUM LOW.

Division of fuzzy sets can take on the property of performing a relative order of magnitude calculation. Here, it is assumed that the median fuzzy set approximates the equivalent numerical value of one (see Figure 6). Therefore, those fuzzy sets below the median are treated as if numerically they are between zero and one, while those fuzzy sets above the median are treated as greater than one. Hence, if a large magnitude number is divided by another large magnitude number, the result is somewhere near one. If a large magnitude number is divided by a very small magnitude number, the result is an even larger magnitude number. Conversely, a small magnitude number divided by a large magnitude number results in an even smaller magnitude number than the original.

3.6 Scalability

An issue that plagues many algorithms is scalability. In the case of the statistical analysis approach outlined on page 29, the equations scale linearly with regard to the data set size. For the Schmucker method, as well as the Kaufmann and Gupta method, the equations do not scale well. In both of these methods, the equations will cause exponential growth in the size of the fuzzy set resulting from the fuzzy arithmetic operations. For the table lookup method, the equations remain linear regardless of the data set size.

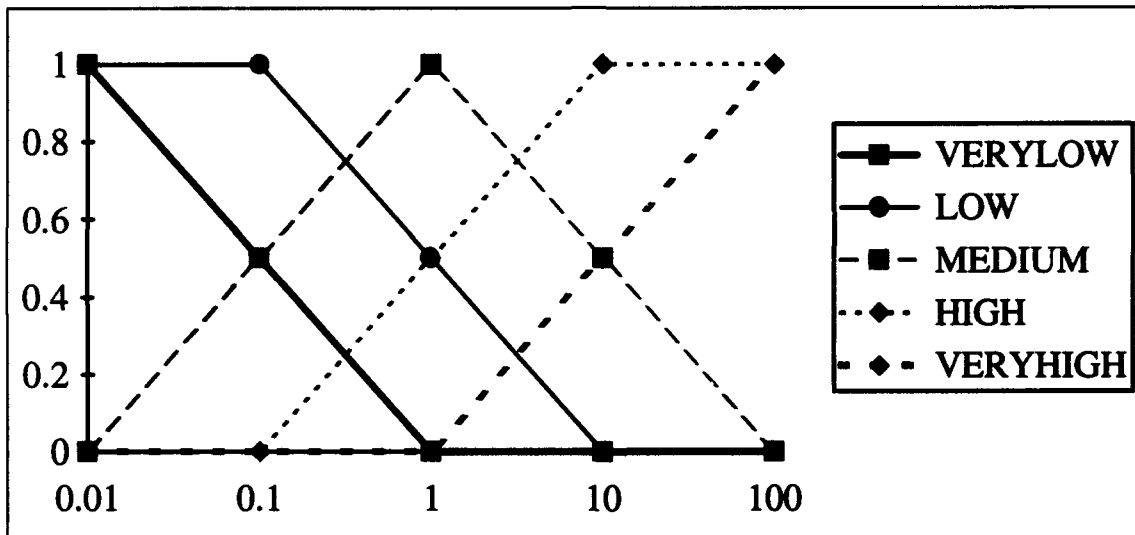


Figure 6. Relative Magnitude Fuzzy Sets.

3.7 Sample Data Generation

In order to test the methods, sample data had to be generated. The output generated by ARES version 2.0 was used as a source of possible vulnerabilities. ARES was used for

convenience, but a random sample of vulnerabilities could be generated by hand. The vulnerabilities output by ARES were combined into a single file. These vulnerabilities are in the form of text strings such as "The system does not use passwords."

This list of vulnerabilities was then put into a LISP object class structure as shown in Figures 7 and 8. Seven object slots, one for each functional area being considered, are associated with each vulnerability object. Each slot value represents how the vulnerability is allocated to each functional area and can have a value in the inclusive range of 0 to 1 for the numeric case. These values were randomly generated. For the linguistic case, a term is used to represent the approximate distribution. The linguistic terms implemented were VERYLOW, LOW, MEDIUMLOW, MEDIUM, MEDIUMHIGH, HIGH, and VERYHIGH. Each of these terms is subject to the translation mapping discussed in the previous sections.

3.8 Summary

This chapter discussed the qualitative and quantitative methods used in this thesis. The quantitative method uses statistical analysis techniques familiar to most readers, while the qualitative methods use fuzzy arithmetic to perform intermediate calculations such as influence contribution. Two methods to model fuzzy arithmetic are given, Schmucker's methods and a table lookup method based on behavior grouping. There is a preference for the table lookup method because of its linear characteristics with regard to the number of vulnerabilities being processed. Finally, this chapter discussed how the vulnerability data is generated.

```
(setf V100 (make-instance 'vuln-node
                        :vuln "The system does not have audit trails. "
                        :code-name 'V100
                        :vuln-influence 0.7334
                        :dist-audit 0.6833
                        :dist-recover 0.4680
                        :dist-access 0.5881
                        :dist-media 0.1887
                        :dist-os 0.8490
                        :dist-configuration 0.4732
                        :dist-documentation 0.5845
                        :vuln-present t
                        ))
```

Figure 7. Lisp Object with Numerical Data

```
(setf F100 (make-instance 'vuln-node
                        :vuln "The system does not have audit trails. "
                        :code-name 'F100
                        :vuln-influence 'H
                        :dist-audit 'MH
                        :dist-recover 'M
                        :dist-access 'MH
                        :dist-media 'L
                        :dist-os 'H
                        :dist-configuration 'M
                        :dist-documentation 'MH
                        :vuln-present t
                        ))
```

Figure 8. Lisp Object with Fuzzy Data

IV Implementation

4.1 Introduction

In this chapter, I will discuss the actual implementation details for two of the analysis methods identified in the previous chapter. Again, because of the lack of an industry standard, it was left to my discretion as to how and what to implement. I felt that with simpler tools, the comparison between methods would be clearer. Therefore, I implemented a quantitative method using statistical analysis and a qualitative method using fuzzy logic.

The code for this thesis was implemented on a Sun SPARCstation 2+ running SunOS 4.0.x. The code was implemented using Sun Common Lisp 4.0 with CLOS extensions. The primary purpose of using Lisp was the ease with which long linked-list structures are handled. Both the numerical and fuzzy logic implementations used the same CLOS object structure as shown in Figure 9.

4.2 Quantitative Method using Statistical Analysis

The quantitative method implemented a small subset of statistical measures. These statistical measures were weighted averaging and standard deviation. These measures were deemed adequate to demonstrate the type of information derivable from the data. Other more complex measures could have been implemented, but it was felt they would simply complicate the comparison between methods and did not appear to provide additional information.

```

(defclass vuln-node ()
  ((vuln
    :initform '()
    :initarg :vuln
    :accessor vuln
    :documentation "Text string with name of vuln")
   (code-name
    :initform '()
    :initarg :code-name
    :accessor code-name
    :documentation "code for internal assignment purposes")
   (vuln-influence
    :initform 0.0
    :initarg :vuln-influence
    :accessor vuln-influence
    :documentation "weight of this vulnerability to the whole")
   (dist-audit
    :initform 1.0
    :initarg :dist-audit
    :accessor dist-audit
    :documentation "Degree with which vuln affects audit")
   (dist-recover
    :initform 0.0
    :initarg :dist-recover
    :accessor dist-recover
    :documentation "Degree with which vuln affects recovery")
   (dist-access
    :initarg :dist-access
    :initform 0.0
    :accessor dist-access
    :documentation "Degree with which vuln affects access")
   (dist-media
    :initform 0.0
    :initarg :dist-media
    :accessor dist-media
    :documentation "Degree with which vuln affects media control")
   (dist-os
    :initform 0.0
    :initarg :dist-os
    :accessor dist-os
    :documentation "Degree with which vuln affects operating system")
   (dist-configuration
    :initform 0.0
    :initarg :dist-configuration
    :accessor dist-configuration
    :documentation "Degree with which vuln affects configuration")
   (dist-documentation
    :initarg :dist-documentation
    :initform 0.0
    :accessor dist-documentation
    :documentation "Degree with which vuln affects documentation")
   (vuln-present
    :initform t
    :initarg :vuln-present
    :accessor vuln-present
    :documentation "Is the vulnerability present")
  ))

```

Figure 9. CLOS Object Structure

To actually determine the contribution of a vulnerability to a functional area, the product of the vulnerability influence and the distribution value for that functional area was calculated. Then based on these contributions, the average contribution and standard deviation were calculated for the data sets. The contribution values were then sorted and the vulnerabilities in or matching the top 10% of the contributors were identified as critical. The possibility exists for a larger number than 10% to be identified as critical. This would occur as a result of contribution values tying for inclusion in the top 10%. Since there is no reason to discriminate against these value based solely on position within the sorted contributions, they are included in the critical vulnerabilities.

The vulnerabilities were also grouped by units of standard deviation. It should be pointed out that the 10% value and the use of standard deviation are arbitrary thresholds. It was necessary to establish some threshold in order to determine critical vulnerabilities.

4.3 Qualitative Method using Fuzzy Logic

4.3.1 Fuzzy Math. In order to make comparison between the methods, the same type of information generated by the quantitative method was desired. To calculate the influence contributions, it is necessary to multiply the influence value for each vulnerability by the distribution values given. This was done using a table lookup. The lookup tables for the addition and multiplication of fuzzy values are given in Tables 1 and 2 respectively.

As mentioned in the previous chapter, calculation of a weighted average is impossible with linguistic terms. In order to calculate an average contribution for each functional area, the sum of the contributions was normalized by the sum of the vulnerability influences.

Just like in the quantitative method, the vulnerabilities in or matching the top 10% of the contributors were identified as critical along with those vulnerabilities in or matching the top 10% of more than one functional area. Again, ties are handled by being included in the identification of critical vulnerabilities. The vulnerabilities were also grouped by linguistic value in order provide additional insight into the structuring of vulnerabilities for each functional area. This grouping of vulnerabilities was based on linguistic values. Also shown on the output is the number of each linguistic value occurring as a contribution value to a specific functional area. This data is provided to show the range of values occurring within each functional area.

Although I implemented a version of Schmucker's method and attempted to execute this method against a sample of 50 vulnerabilities, this method never successfully completed execution. The problem was not with the implementation, but combinatorial explosion. After running for over 24 hours, the Schmucker method would consistently cause out of memory errors. As such, I was never able to acquire results using this method on a complete sample of vulnerabilities.

As mentioned in the previous chapter, these lookup tables were built based on the majority behavior of the indicated operation. There were a few cases where the majority was only slightly larger than the minority. It is possible because of this to see slightly unexpected behavior if this table lookup method is compared to a numeric method. If the numeric values fall near the mapping boundaries, the linguistic result may be off by at most one category. In no case did the boundary shift by more than one fuzzy term.

4.4 Summary

This chapter described how each of the vulnerability analysis methods were implemented. Also discussed was how the critical vulnerabilities were identified and how the vulnerabilities were

grouped according to units of standard deviation in the quantitative case and linguistic terms in the qualitative case. Finally, how the lookup tables were built and problems with boundary conditions were discussed.

<i>Add</i>	<i>VERYLOW</i>	<i>LOW</i>	<i>MEDLOW</i>	<i>MEDIUM</i>	<i>MEDHIGH</i>	<i>HIGH</i>	<i>VERYHIGH</i>
<i>VERYLOW</i>	VERYLOW	LOW	MEDLOW	MEDIUM	MEDHIGH	HIGH	VERYHIGH
<i>LOW</i>	LOW	MEDLOW	MEDIUM	MEDHIGH	HIGH	VERYHIGH	VERYHIGH
<i>MEDLOW</i>	MEDLOW	MEDIUM	MEDHIGH	HIGH	VERYHIGH	VERYHIGH	VERYHIGH
<i>MEDIUM</i>	MEDIUM	MEDHIGH	HIGH	VERYHIGH	MEDHIGH	VERYHIGH	VERYHIGH
<i>MEDHIGH</i>	MEDHIGH	HIGH	VERYHIGH	MEDHIGH	VERYHIGH	VERYHIGH	VERYHIGH
<i>HIGH</i>	HIGH	VERYHIGH	VERYHIGH	VERYHIGH	VERYHIGH	VERYHIGH	VERYHIGH
<i>VERYHIGH</i>	VERYHIGH	VERYHIGH	VERYHIGH	VERYHIGH	VERYHIGH	VERYHIGH	VERYHIGH

Table 1. Addition of Fuzzy Sets (Linear Mapping)

<i>Multiply</i>	<i>VERYLOW</i>	<i>LOW</i>	<i>MEDLOW</i>	<i>MEDIUM</i>	<i>MEDHIGH</i>	<i>HIGH</i>	<i>VERYHIGH</i>
<i>VERYLOW</i>	VERYLOW	VERYLOW	VERYLOW	VERYLOW	VERYLOW	VERYLOW	VERYLOW
<i>LOW</i>	VERYLOW	VERYLOW	VERYLOW	VERYLOW	VERYLOW	LOW	LOW
<i>MEDLOW</i>	VERYLOW	VERYLOW	VERYLOW	LOW	LOW	LOW	MEDLOW
<i>MEDIUM</i>	VERYLOW	VERYLOW	LOW	LOW	MEDHIGH	MEDLOW	MEDIUM
<i>MEDHIGH</i>	VERYLOW	VERYLOW	LOW	MEDHIGH	MEDLOW	MEDIUM	MEDHIGH
<i>HIGH</i>	VERYLOW	LOW	LOW	MEDLOW	MEDIUM	MEDHIGH	HIGH
<i>VERYHIGH</i>	VERYLOW	LOW	MEDLOW	MEDIUM	MEDHIGH	HIGH	VERYHIGH

Table 2. Multiplication of Fuzzy Sets (Linear Mapping)

V Results

5.1 Introduction

This chapter will outline the results obtained from the methods and implementations given in Chapters 3 and 4. Although the Schmucker method was implemented, that method will not be used for comparison purposes due to its significant problem with scalability. The discussion of the comparison results will only be based on the quantitative analysis method using statistical analysis and the qualitative analysis method using the table lookup of fuzzy arithmetic functions.

Given that there is no industry standard with which to compare these results, some of the comparisons given below have to be subjective in nature. When a subjective comparison is made, an attempt is made to explain the basis for the comparison and how the results were interpreted.

5.2 Hypothesis (restated)

The hypothesis of this thesis is stated in two parts: that a qualitative analysis approach to vulnerability analysis is as effective and efficient as a quantitative approach and that the qualitative approach provides the security analyst with intuitive information not readily available in quantitative approaches. Effectiveness is the ability to provide reasonable categorizations of identified vulnerabilities based on influence contributions to a functional area. It is measured with regard to a method's ability to categorize vulnerabilities into reasonable clusters and how easy it is for that method to be used. Efficiency is measured with regard to processing time and scalability of the method.

5.3 Effectiveness

When evaluating any method to perform a specific task, it is essential to determine the effectiveness of that method in performing the task. For computer security vulnerability analysis, a method is effective if it can categorize or cluster the vulnerabilities into reasonable groupings based on importance or impact. This ability is necessary, but it is not sufficient for determining effectiveness. The ease with which the method can be applied must also be considered. A method may be very effective at performing categorizations, but if it requires extensive data setup or the results of the method are difficult to interpret, most would agree that the method loses its effectiveness.

5.3.1 Ability to Categorize Vulnerabilities The measurement of the ability of a given method to categorize vulnerabilities is divided into four main concerns: accuracy, precision, generation of distinctions, and evaluation of close results. Most of these concerns are evaluated on a subjective basis and the evaluation may depend on the specific application of the method.

5.3.1.1 Accuracy. A concern with any method is the accuracy of the results. A true determination of accuracy requires that an accepted method be the baseline to compare the results of other methods against. This presented an insurmountable problem as there is no industry or academic standard method.

However, a subjective determination of accuracy was possible by looking at broad clusterings of the vulnerabilities. In this, I sought to determine if those vulnerabilities that were on the high end of importance scale for one method were also on the high end for the other method. Note that this does not imply that the actual order of vulnerabilities is the same for both methods, only that the broad clusterings were similar. Given this subjective method, the two methods

produced comparable broad clusters (see Figures 10 and 11). Reviewing the data shown in these figures shows that vulnerabilities V100 and V125 and the corresponding vulnerabilities F100 and F125 were clustered in the top category. Likewise, vulnerabilities V121 and V126 and the corresponding F121 and F126 were clustered in the lowest category. I cannot say, nor is it possible to without a baseline standard, that one method is more accurate than the other. I can say, that based on the subjective broad clusterings, each method appears to have comparable accuracy.

Functional Area: AUDIT									
Sigma Group: 1	==>	V100	V125	V131					
Sigma Group: 2	==>	V143	V148	V115	V101	V142	V112		
Sigma Group: 3	==>	V137	V141	V117	V140	V144	V129	V135	V133
		V118	V124	V134	V122				
Sigma Group: 4	==>	V139	V128	V113	V136	V130	V132	V127	V114
		V111	V105	V108	V107	V106	V119	V110	V147
		V138	V120	V145	V103	V146	V116	V102	V149
		V109	V104	V123	V126	V121			

Figure 10. Clustering of Audit Functional Area (Quantitative)

Functional Area: AUDIT									
Importance: M	==>	F125	F100						
Importance: ML	==>	F101	F112	F131	F143	F148			
Importance: L	==>	F122	F142	F115	F124	F144	F137	F141	F117
		F133	F135	F118	F129	F134	F140		
Importance: VL	==>	F127	F106	F113	F108	F128	F132	F110	F136
		F139	F105	F111	F114	F119	F102	F109	F130
		F107	F103	F138	F146	F104	F116	F120	F147
		F123	F126	F145	F149	F121			

Figure 11. Clustering of Audit Functional Area (Qualitative)

5.3.1.2 Precision. Another concern in evaluating the effectiveness of a method is precision. Precision differs from accuracy and the two should not be confused. Accuracy

implies a degree of correctness, whereas precision is the degree of resolution. For instance, 3.14 and 3.1459265 are both estimates of the value of pi. Neither is completely accurate, but the second value is more precise. So is 1.5923111, but it is obviously less accurate. This illustrates that high precision does not imply high accuracy.

In evaluating the precision of the two methods, most would agree that a numeric solution would have a higher degree of precision over a non-numeric solution. This can be seen in that the numeric solution has, at least theoretically, an infinite degree of precision, while the non-numeric solution is limited to the resolution provided by the linguistic terms.

Since the numeric solution is more precise, it could be inferred that this added precision also adds information to the results and as such, this added information should be usable. The problem with making this inference is that the original input to the problem, the subjective assignment of influence and distribution for each vulnerability, lacks precision.

In the numeric method, the analyst has an infinite range of values between 0 and 1 that can be assigned to each influence or distribution value. While this allows the analyst to provide a higher degree of resolution of the input, it does not necessarily add information to the analysis. For example, the analyst is providing influence values to two vulnerabilities, and while they both have a subjective rating of medium, the analyst wants one to be slightly more medium than the other. She therefore assigns one an influence value of 0.51 and the other an influence value of 0.55. Also assume that for a given functional area, both vulnerabilities have a distribution value of 1.0. When all of the calculations are performed and the vulnerabilities have been clustered, it is possible that the two vulnerabilities will fall into the same cluster.

In reality, providing this level of precision on the input has not affected the overall assignment of the vulnerabilities to a specific cluster. Likewise, it is possible that increasing the resolution of the output values will not change the clustering. What increasing the precision in the

data input and the results output would do is provide a false sense of improved accuracy. The case where the two vulnerabilities do fall into different clusters will be discussed below concerning evaluation of close results.

The level of precision provided by a method is directly affected by the true precision of the data input into that method. If the analysis starts with inherently imprecise data, it is not reasonable that the true accuracy of the results will increase just because the data is expressed in terms of increased precision. While it is true that the numeric method has a greater possible precision due to increased resolution of the input values, I contend that this increased precision simply leads to a false assumption that the results are more accurate. Simply put, subjective inputs lead to subjective outputs and precise subjective inputs lead to precise subjective outputs. The key point is that the outputs, regardless of how precise the inputs, is still subjective.

5.3.1.3 Generation of Distinctions. In performing the categorization of vulnerabilities, it is necessary to generate distinctions among data values. By generating these distinctions, the method is able to group the information into clusters. *The concern here is how effective are the two methods at generating these distinctions.*

In the non-numeric case, the most obvious distinction is by linguistic value. Grouping the analysis results by linguistic values provides an intuitive clustering of the information. For instance, it make sense to group all of the vulnerabilities with a VERYHIGH influence contribution together, then the vulnerabilities with a HIGH influence contribution, and so forth. Even within these clusters, further distinctions can be made based on linguistic values by grouping the vulnerability influence values that are the same or by grouping the vulnerability distribution values for a specific functional area. The distinctions are appropriate given the desired goal of trying to cluster all of the vulnerabilities by importance.

In the numeric case, there is no obvious distinction. Any distinction made is arbitrary since the range of possible values is infinite. Some possible distinctions that could be made would include grouping the data by units of standard deviation (the choice used in this thesis), developing confidence intervals, or calculating histogram clusterings. There are a plethora of statistical methods that could be used to cluster the vulnerability data, but none is sufficient for all circumstances. This is one of the primary reasons that no industry standard method has been developed.

Again, it must be stressed that the original input values to either method are subjective and as such the output of either method is subjective. Any distinctions made between adjacent data points in the numeric method are arbitrary.

For instance, assume an university chooses its distinguished graduates from the students within the top ten percent of the graduating class's grade point average (GPA). The cutoff, as determined by numeric methods is 3.9781. All students whose GPA is 3.9781 and greater are classified as distinguished graduates. Assume another student has a GPA of 3.9780. Using a strictly numeric cutoff eliminates this student from being a distinguished graduate. Also note, that if the precision of the cutoff was reduced to 3.978, and all GPAs rounded to three digits, the student would be selected. Of course, this selection of three significant digits is just as arbitrary as four digits and the same boundary condition occurs for someone with a GPA of 3.9774.

The purpose of this example is to show how placing an arbitrary and fixed cutoff to the data being used can lead to undesired results. In performing the distinguished graduate selection, the university's real goal was to reward those students whose academic achievement place them in or *very near* the top ten percent of their class.

5.3.1.4 Evaluation of Close Results. The examples given in the previous section bring to light the problem of how to evaluate close results. Close results are those results

that fall on or near the distinction cutoffs being used. In the case of non-numeric method, it is adequate to say that a result is close to another if they have the same linguistic values. This is adequate because the linguistic values represent ranges of values and these ranges are fairly large. This has the effect of enforcing the desired behavior that if two vulnerabilities are close in importance, then they should be considered together and it is not reasonable to distinguish between them.

In the numeric case, as was demonstrated by the above examples, two values can be numerically close and still fall into different clusters. This is true even if a histogrammatic program is used to try and determine the 'natural' clusters. This is caused by the exactness of the numeric method where each value used is in effect its own cluster. Because of the infinite number of individual clusters, the numeric method will apply the chosen arbitrary cutoff between two vulnerabilities. In reality, these two vulnerabilities should be considered together, but the numeric method may be unable to detect this closeness and arbitrarily applies the cutoff.

5.3.2 *Ease of Use* Ease of use can only be measured subjectively. What is easy for one person to use, may be difficult for another person. However, even with this in mind, there are certain attributes of each method that are demonstrative of their ease of use. The most important measure of ease of use is the ease of interpreting the data.

5.3.2.1 Interpretation of Data. As with any product, the main goal is to make meaningful and consistent interpretations of the output results. A consistent interpretation of the results implies that a single user construes the same information from the results every time the results are reviewed. A consistent interpretation also implies that there lacks ambiguity in what the results mean. A meaningful interpretation is much harder to define, but can be seen as whether the analyst must try and guess what the meaning of the values output represent. Another definition of meaningful interpretation could be whether the data has an implied semantic meaning.

The first interpretation of data is made with regard to the input data. It is essential that the input data be generated and interpreted consistently. Assume for the numeric case, the user is asked to assign an influence value to a vulnerability with the possible choices being between 0 and 100, and the user selects 56. Would there have been any significance to choosing 56 over 57? Not in reality, unless the equations used to perform the calculations are extremely sensitive to small changes in input. As such a sensitive system would probably not be fielded in the first place, I assert there is no significant difference in choosing 56 over 57 in the preceding question.

Some might argue that the user has too many choices, so the choices are lowered from 0 to 100 to the range 0 to 9. Here we would expect that there is a significant difference between 5 and 6. Lowering the number of choices improved the resolution of choice significance. What then is the significance of the user assigning 5 (from the choice of 0 to 9) to an influence value? This could now be interpreted to mean that the vulnerability has a medium influence value. I contend that the user should input the value **MEDIUM** that conveys the semantic meaning intended by the user.

The same semantic difficulties arise in interpreting system output. After processing all of the identified vulnerabilities using whatever numeric methods chosen, the system generates an overall vulnerability rating of 8 from the range of 0 to 9. This could be interpreted to mean a high vulnerability rating if the value of 9 represents the high end of the scale. It could also be interpreted to mean a low rating if 9 represents the low end of the rating scale. If on the other hand, the system indicates that the overall vulnerability rating is **HIGH**, there is little left to interpret.

In the numeric case, the user is presented with a list of statistical values that attempt to represent how the vulnerabilities are important to each functional area. In most cases, the analyst makes a internal conversion from the numeric values to their intended linguistic meaning. Internal

conversion implies that the values are converted in the analyst head as opposed to a algorithm implemented on a computer. The analyst would look at the data and build internal subjective scales to evaluate the meaning of the numeric values. The inconsistency arises when a few days, months, or years later, the same analyst reviews the output results. It is up to the analyst to try and remember the exact interpretation made when the results were previously reviewed.

There can also be inconsistencies when multiple analysts are reviewing the numerical information. It is very unlikely that any two analysts would generate the same internal scales to evaluate the meaning of the numeric values.

In the non-numeric case, there is a fairly standard meaning associated with each of the linguistic terms. These meanings are fairly standard in that most security analysts would have similar internal representations of the concepts HIGH, MEDIUM, and LOW. There may be some latitude in the specific meanings associated with compound linguistic terms such as VERYLOW and MEDIUMHIGH, but the general meanings associated with these types of terms should be consistent within a particular domain.

Looking at the non-numeric data, the analyst is not left to speculate as to whether a vulnerability has a HIGH or VERYHIGH rating; the data simply states the value. The analyst is also not required to remember from day to day, any numeric range of values that constitute a particular linguistic term.

It is in this category of ease of interpretation of data that the non-numeric method excels the greatest over the numeric method.

5.4 Efficiency

5.4.1 Timed Performance. The timed performances for each of the methods were comparable. The actual times, in seconds, are shown in Table 3 and plotted in Figure 12. The performance was almost identical for the 50 vulnerability sample, while the numerical method was better for the 120 data sample. The 198, 250, 320, and 400 sample data sets are combinations of the smaller sets generated only for the purpose of timing. The data shows that the two methods perform comparably and therefore, for vulnerability analysis, neither method is a clear winner.

5.4.2 Scalability. As mentioned in previous chapters, both methods should demonstrate linear scalability. This is demonstrated by Figure 12. It should be noted that a portion of the time shown in Table 3 can be attributed to programming overhead that is not related to data set size. Given the linearity of each method, either would be appropriate for vulnerability analysis.

5.5 Summary

The above results demonstrate that both methods perform equally well with regard to timed performance, and scalability. However, there are significant differences in the ability of the two methods to categorized the vulnerability information and the ease of which the data can be interpreted.

For the average user, the fuzzy analysis method has the advantage here. The thought process associated with assigning influence values and how these influence values are distributed is much more natural with the fuzzy logic approach. Also, the analysis of the end product is more

intuitive when using linguistic terms as opposed to numerical values. This advantage is born out by a comment included in Schmucker:

A higher degree of response consistency over trials was found to occur if the subject is allowed to give an imprecise verbal response about a fuzzy (concept) than if he is forced to give a precise "grade-of-membership answer [Kochen, 1975]. (Schmucker84:35)

5 Runs Per Set (size)	(50)	(120)	(198)	(250)	(320)	(400)
Fuzzy Best	0.69	1.60	2.96	3.62	4.84	6.40
Fuzzy Worst	0.76	1.66	3.20	3.68	4.89	6.50
Fuzzy Average	0.72	1.63	3.08	3.64	4.87	6.45
Numeric Best	0.68	1.12	3.12	3.64	4.32	6.78
Numeric Worst	0.77	1.19	3.59	3.95	4.47	7.06
Numeric Average	0.72	1.16	3.40	3.76	4.39	6.90

Table 3. Timed performance (Numerical vs. Fuzzy)

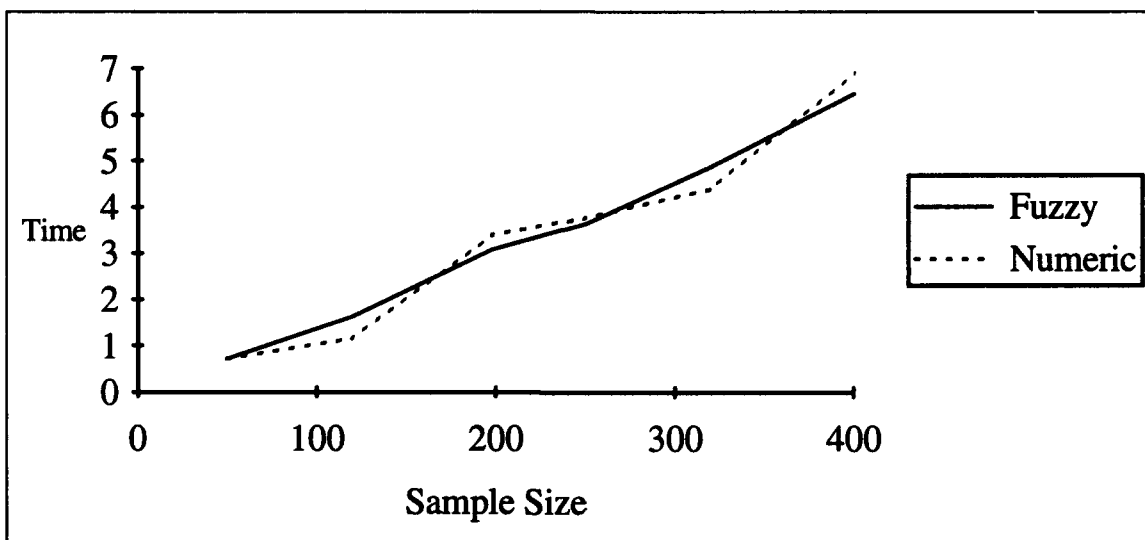


Figure 12. Timed performance (Numerical vs. Fuzzy)

VI Conclusions and Recommendations

6.1 Conclusions

This thesis has demonstrated the feasibility of using a non-traditional method to process and analyze computer security vulnerabilities. It has demonstrated that not only is this non-traditional method comparable to a similarly structured quantitative method, but with regard to ease of use, better. Although fuzzy logic was the non-traditional method used in this thesis, other methods, such as those listed in chapter 2, could possibly be applied.

The use of linguistic terms allows the end user to make meaningful evaluations of vulnerability influence and impact. Unlike numbers, a linguistic term has a semantic meaning as well as representing a quantity.

6.2 Project Recommendations

This thesis recommends the use of a qualitative analysis method to perform vulnerability analysis. This recommendation is based on the ease of use and intuitive nature of qualitative analysis methods such as fuzzy logic.

6.3 Future Enhancements and Phases

A future enhancement and continuation of this work would be to create an inference mechanism to combine the results of each functional area. This inference mechanism could be in

the form of rules that would direct how to interpret the results. The final output of such an inference mechanism could be a single vulnerability rating or score.

As the first phase of a multi-phase research project, this thesis lays the groundwork for building a fully automated computer security risk analysis system. In addition to investigating and implementing the recommendations listed in the following section, future phases of the research project include addressing implementation tradeoffs between risk analysis capabilities and host architecture constraints (e.g., memory limitations and microprocessor performance), developing adaptive analysis techniques to allow for and take advantage of new technology, and generation of new reasoning with uncertainty methods.

This thesis has addressed only one specific part of automating computer security: specifically, whether a qualitative or quantitative analysis method is more applicable to vulnerability analysis. As the use of automated methods to evaluate computer security is still in its infancy, the possibilities for further research are almost limitless. The following paragraphs suggest a few of these areas where additional work could, and probably should, be done.

The vulnerability assessment methods explored in this thesis could be expanded to include combinations of multiple vulnerabilities. This thesis dealt with evaluating individual vulnerabilities that were independent of each other. Future research should be conducted which will allow for combinations of vulnerabilities to be assessed. An example would be a system that lacks passwords and the facility where the system is located is not locked. Each of these alone has a vulnerability rating associated with it, but the combined effect of both vulnerabilities may have a much higher rating than either alone.

Another area that should be explored is the generation of other non-numeric analysis methods that provide linear, or very near linear, performance. Along with this research would be analysis into the number of linguistic terms necessary to adequately represent the range of possible

vulnerability levels and analysis into how the size and structure of the lookup table affects predictability of results.

Effort should be expended towards the development of an expert system that is capable of making the necessary recommendations for vulnerability correction and threat elimination. Currently, the USAF depends on experts at AFCSC and the CSOs to determine the necessary steps to correct vulnerabilities and eliminate threats. Codifying the expertise of these individuals into a portable expert system would allow new and less experienced CSOs to provide enhanced and reliable protection of their resources.

Work could be done on using hardware and software design specifications to determine security flaws. This would allow for security flaws to be corrected prior to the system being implemented. Most hardware and software systems are built with performance and ease of use as driving factors. Security is often not considered until it is too late to correct a flaw. Using information from the specification and design phases, an assessment could be performed to identify possible security weaknesses prior to the system being built. This earlier assessment would allow the system to be built with most if not all of the security flaws corrected.

Future work could be put forth on developing a system capable of performing threat assessment to determine if known threats could penetrate the current safeguards. This research would involve building a simulation model of each system's safeguards and then applying the known threats to the model. As a continuation of the threat assessment concept, automatic threat scenario generation could be developed that would generate a possible sequence of events leading to the compromise of a computer system. The only drawback to this type of research would be the sensitivity. It's obvious that any system that generates and evaluates threat scenarios would have to be highly classified.

Appendix A: Computer Programs

A-1 Quantitative Method using Statistical Analysis

The following code excerpts are provided to illustrate high level functionality of the implemented code.

Functions used to generate statistics

```
(defun sum-em1 (slot-name)
  "Calculate influence values for a particular functional area"
  (setf sum1 0)
  (mapc #'(lambda (x)
    (progn
      (cond
        ((vuln-present (eval x))
         (setf value (get-value (eval x) slot-name))
         (setf value2 (vuln-influence (eval x)))
         (setf sum1 (+ sum1 (* value value2))))
        (t (setf sum1 (+ 0 sum1))))))
    vuln-list)
  sum1)

(defun sum-em2 (slot-name)
  "Calculate the squared influence values for a particular functional area"
  (setf sum2 0)
  (mapc #'(lambda (x)
    (progn
      (cond
        ((vuln-present (eval x))
         (setf value (get-value (eval x) slot-name))
         (setf value2 (vuln-influence (eval x)))
         (setf sum2 (+ sum2 (expt (* value value2) 2 ))))
        (t (setf sum2 (+ 0 sum1))))))
    vuln-list)
  sum2)

(defun sum-squared ()
  "Initialize variable for sum of each slot"
  (setf ssum (mapcar #'sum-em2 dist-slots)))

(defun rawsum1 ()
  "Initialize variable for sum of each slot"
  (setf inf-sum (sum-em 'vuln-influence))
  (setf areas (mapcar #'sum-em1 dist-slots))
  (setf sum-areas (apply #' + areas))
  (setf areas-pct (mapcar #'(lambda (x) (/ x sum-areas)) areas))
  (setf sum-areas-pct (apply #' + areas-pct)))
```

```

areas)

(defun calc-inf ()
  "Calculate the influence for each functional area"
  (setf inf-calculated t)
  (preponderance) ; generate list of # vuln present in each area
  (rawsum1) ; generate raw weighted sums
  (setf area-avgs (mapcar '/ areas prepond))
  (sum-squared)
  (setf variance (mapcar #'(lambda (n x x2)
    (progn
      (* (/ 1 (1- n))
        (- x2 (* (/ 1 n)
          (expt x 2))))))
    prepond areas ssum))
  (setf stdev (mapcar #'sqrt variance))
  )

```

Functions used to generate clusterings

```

(defun count-sd-groups (slot-name)
  (cond
    ((eval inf-calculated) nil)
    (t (calc-inf) (gen-sorted-inf-list)))
  (setq posv (position slot-name dist-slots))
  (setq maxvuln (car (nth posv sorted-inf-list)))
  (setq maxval (* (get-value (eval maxvuln) slot-name)
    (vuln-influence (eval maxvuln))))
  (setq sd (nth posv stdev))
  (multiple-value-setq (sdgroups junk) (ceiling (/ maxval sd)))
  sdgroups)

(defun group-sd-vals (slot-name)
  (setq groups (count-sd-groups slot-name))
  (setq posv (position slot-name dist-slots))
  (setq maxvuln (car (nth posv sorted-inf-list)))
  (setq maxval (* (get-value (eval maxvuln) slot-name)
    (vuln-influence (eval maxvuln))))
  (setq sd (nth posv stdev))
  (setf sd-groups (list maxval))
  (loop for num from 2 to groups
    do
      (progn
        (setf minval (- maxval sd))
        (cond
          ((< minval 0) (setf minval 0)) (t nil))
        (setf sd-groups (append sd-groups (list minval)))
        (setf maxval minval)))
  sd-groups)

(defun count-sd-groups-content (slot-name)
  (setq numgrps (count-sd-groups slot-name))
  (setq grprngs (group-sd-vals slot-name))
  (setq array (make-list numgrps :initial-element 0))
  (setq posf (position slot-name dist-slots))
  (setq temp-sorted-list (copy-list (nth posf sorted-inf-list)))
  (loop for pos1 from 0 to (1- (length temp-sorted-list))
    do
      (let*

```

```

        ((temp-vuln (nth pos1 temp-sorted-list))
         (value1 (get-value (eval temp-vuln) slot-name))
         (value2 (vuln-influence (eval temp-vuln)))
         (value3 (* value1 value2))
         (posx (position value3 grprngs :test #'< :from-end t)))
      )
    (cond
      ((null posx) nil)
      (t (setf (nth posx array) (1+ (nth posx array)))))
  ))
array)

(defun group-sd-content (slot-name)
  (setq groups (count-sd-groups slot-name))
  (setq array1 (make-list groups :initial-element nil))
  (setq posf (position slot-name dist-slots))
  (setq ts1 (copy-list (nth posf sorted-inf-list)))
  (setq total-pos 0)
  (setq llist (count-sd-groups-content slot-name))
  (setq lfuz (1- groups))
  (loop for pos1 from 0 to lfuz
    do
      (let*
        ((inc (nth pos1 llist))
         (start total-pos)
         (stop (+ total-pos inc))
         )
        (setf (nth pos1 array1) (subseq ts1 start stop))
        (setf total-pos (+ total-pos inc))
      ))
  array1)

(defun top-portion (slot-name count)
  (setq cnt count)
  (setq loop-stop 0)
  (cond
    ((eval list-inf-generated) nil)
    (t (gen-sorted-inf-list)))
  (let*
    ((posf (position slot-name dist-slots))
     (ftsl (copy-list (nth posf sorted-inf-list)))
     (last (nth (1- count) ftsl))
     (infx (get-value (eval last) 'vuln-influence))
     (valx (get-value (eval last) slot-name))
     (prodx (* infx valx)))
    (loop while (eq loop-stop 0)
      do
        (let*
          ((next (nth cnt ftsl))
           (infx (get-value (eval next) 'vuln-influence))
           (valy (get-value (eval next) slot-name))
           (prody (* infx valy)))
          (cond
            ((eql prody prodx) (setf cnt (1+ cnt)))
            (t (setf loop-stop 1))))
        (subseq ftsl 0 cnt)))

(defun top-contrib ()
  (preponderance)
  (setq pcnt (floor (* (car prepond) 0.10)))
  (setq all-c (mapcar #'(lambda (x) (top-portion x pcnt)) dist-slots))
  (setq candidates (sort
    (remove-duplicates (flatten all-c))
    #'string-lessp))

```

```

(setq withdups (flatten all-c))
(setq candcount (mapcar #'(lambda (x)
                             (count x withdups)) candidates))
(setq lpos (1- (length candidates)))
(setq tc (make-list 7 :initial-element candidates))
(loop for pos from 0 to lpos
  do
    (setq tmpval (nth pos candidates))
    (case (nth pos candcount)
      ((7) t)
      ((6) (setf
              (subseq tc 6 7)
              (mapcar
               #'(lambda (x) (remove tmpval x)) (subseq tc 6 7))))
      ((5) (setf
              (subseq tc 5 7)
              (mapcar
               #'(lambda (x) (remove tmpval x)) (subseq tc 5 7))))
      ((4) (setf
              (subseq tc 4 7)
              (mapcar
               #'(lambda (x) (remove tmpval x)) (subseq tc 4 7))))
      ((3) (setf
              (subseq tc 3 7)
              (mapcar
               #'(lambda (x) (remove tmpval x)) (subseq tc 3 7))))
      ((2) (setf
              (subseq tc 2 7)
              (mapcar
               #'(lambda (x) (remove tmpval x)) (subseq tc 2 7))))
      ((1) (setf
              (subseq tc 1 7)
              (mapcar
               #'(lambda (x) (remove tmpval x)) (subseq tc 1 7))))
    ))
  t)

```

A-2 Qualitative Method using Fuzzy Analysis (Table Lookup)

The following code excerpts are provided to illustrate high level functionality of the implemented code.

Functions used to generate statistics

```
(defun sum-em1 (slot-name)
  "Calculate influence values for a particular functional area"
  (setf inf-sum (sum-em 'vuln-influence))
  (setf sum1 'v1)
  (mapc #'(lambda (x)
    (progn
      (cond
        ((vuln-present (eval x))
         (setf value (get-value (eval x) slot-name))
         (setf value2 (vuln-influence (eval x)))
         (setf value3 (divf value2 inf-sum))
         (setf sum1 (addf sum1 (multf value value3))))
        (t (setf sum1 (addf 'v1 sum1))))))
    vuln-list)
  sum1)

(defun sum-em2 (slot-name)
  "Calculate the squared influence values for a particular functional area"
  (setf sum2 'v1)
  (mapc #'(lambda (x)
    (progn
      (cond
        ((vuln-present (eval x))
         (setf value (get-value (eval x) slot-name))
         (setf value2 (vuln-influence (eval x)))
         (setf sum2 (addf sum2 (multf (multf value value2)
                                       (multf value value2))))))
        (t (setf sum2 (addf 'v1 sum1))))))
    vuln-list)
  sum2)

(defun sort-em1 (slot-name)
  "Returns a list of nodes sorted by influence given for a given slot"
  (setf current-sort-slot slot-name)
  (setf sort-list (copy-list vuln-list))
  (sort sort-list #'sort-order-inf-value))

(defun sort-order-inf-value (x y)
  (let*
    ((valx (get-value (eval x) current-sort-slot))
     (valy (get-value (eval y) current-sort-slot))
     (infx (vuln-influence (eval x)))
     (infy (vuln-influence (eval y))))
```

```

(pvx (position valx fuzzy-values))
(pvy (position valy fuzzy-values))
(pix (position infx fuzzy-values))
(piy (position infy fuzzy-values))
(posx (position (multf infx valx) fuzzy-values))
(posy (position (multf infy valy) fuzzy-values))
(cond
  ((> posx posy) t)
  ((and (= posx posy) (> pix piy)) t)
  ((and (= posx posy) (= pix piy) (> pvx pvy)) t)
  ((and (= posx posy) (= pix piy) (= pvx pvy) (string-lessp x y)) t)
  (t nil)))

(defun rawsum1 ()
  "Initialize variable for sum of each slot"
  (setf inf-sum (sum-em 'vuln-influence))
  (setf areas (mapcar #'sum-em1 dist-slots))
  areas)

```

Functions used to generate clusterings

```

(defun count-fuzzy-content (slot-name)
  (setq farray (list 0 0 0 0 0 0 0))
  (cond
    ((eval list-inf-generated) nil)
    (t (gen-sorted-inf-list)))
  (setq posf (position slot-name dist-slots))
  (setq f-temp-sorted-list (copy-list (nth posf sorted-inf-list)))
  (setq lfuz (1- (length fuzzy-values)))
  (loop for pos1 from 0 to (1- (length f-temp-sorted-list))
    do
      (let*
        ((temp-fvuln (nth pos1 f-temp-sorted-list))
         (fvalue1 (get-value (eval temp-fvuln) slot-name))
         (fvalue2 (vuln-influence (eval temp-fvuln)))
         (f-temp-inf (multf fvalue1 fvalue2))
         (posx (- lfuz (position f-temp-inf fuzzy-values))))
        (setf (nth posx farray) (1+ (nth posx farray)))
        ))
    farray)

(defun group-fuzzy-content (slot-name)
  (setq farray1 (list nil nil nil nil nil nil))
  (cond
    ((eval list-inf-generated) nil)
    (t (gen-sorted-inf-list)))
  (setq posf (position slot-name dist-slots))
  (setq f-tsl (copy-list (nth posf sorted-inf-list)))
  (setq total-pos 0)
  (setq llist (count-fuzzy-content slot-name))
  (setq lfuz (1- (length fuzzy-values)))
  (loop for pos1 from 0 to lfuz
    do
      (let*
        ((inc (nth pos1 llist))
         (start total-pos)
         (stop (+ total-pos inc)))
        (setf (nth pos1 farray1) (subseq f-tsl start stop)))

```

```

        (setq total-pos (+ total-pos inc))
      ))
    farray1)

(defun top-portion (slot-name count)
  (setq cnt count)
  (setq loop-stop 0)
  (cond
    ((eval list-inf-generated) nil)
    (t (gen-sorted-inf-list)))
  (let*
    ((posf (position slot-name dist-slots))
     (ftsl (copy-list (nth posf sorted-inf-list)))
     (last (nth (1- count) ftsl))
     (infx (get-value (eval last) 'vuln-influence))
     (valx (get-value (eval last) slot-name))
     (prodx (multf infx valx)))
    (loop while (eq loop-stop 0)
      do
        (let*
          ((next (nth cnt ftsl))
           (infx (get-value (eval next) 'vuln-influence))
           (valy (get-value (eval next) slot-name))
           (prody (multf infx valy)))
          (cond
            ((eql prody prodx) (setq cnt (1+ cnt)))
            (t (setq loop-stop 1))))
        (subseq ftsl 0 cnt)))

(defun top-contrib ()
  (preponderance)
  (setq pcnt (floor (* (car prepond) 0.10)))
  (setq all-c (mapcar #'(lambda (x) (top-portion x pcnt)) dist-slots))
  (setq candidates (sort
    (remove-duplicates (flatten all-c))
    #'string-lessp))
  (setq withdups (flatten all-c))
  (setq candcount (mapcar #'(lambda (x)
    (count x withdups)) candidates))
  (setq lpos (1- (length candidates)))
  (setq tc (make-list 7 :initial-element candidates))
  (loop for pos from 0 to lpos
    do
      (setq tmpval (nth pos candidates))
      (case (nth pos candcount)
        ((7) t)
        ((6) (setf
          (subseq tc 6 7)
          (mapcar
            #'(lambda (x) (remove tmpval x)) (subseq tc 6 7))))
        ((5) (setf
          (subseq tc 5 7)
          (mapcar
            #'(lambda (x) (remove tmpval x)) (subseq tc 5 7))))
        ((4) (setf
          (subseq tc 4 7)
          (mapcar
            #'(lambda (x) (remove tmpval x)) (subseq tc 4 7))))
        ((3) (setf
          (subseq tc 3 7)
          (mapcar
            #'(lambda (x) (remove tmpval x)) (subseq tc 3 7))))
        ((2) (setf
          (subseq tc 2 7)
          (mapcar

```

```

        #'(lambda (x) (remove tmpval x)) (subseq tc 2 7))))
    ((1) (setf
      (subseq tc 1 7)
      (mapcar
        #'(lambda (x) (remove tmpval x)) (subseq tc 1 7))))
    ))
  t)

(defun calc-inf ()
  "Calculate the influence for each functional area"
  (preponderance) ; generate list of # vuln present in each area
  (rawsum1) ; generate raw weighted sums
  (setf area-avgs (mapcar '(lambda (x) (divf x inf-sum)) areas))
  )

```

The following are the fuzzy math functions used to implement the table lookup

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                                    ;;;
;;;      Define Fuzzy math functions                                  ;;;
;;;                                                                    ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(setf fuzzy-values (list 'vl 'l 'ml 'm 'mh 'h 'vh))
(setf vl "Very Low")
(setf l "Low")
(setf ml "Medium Low")
(setf m "Medium")
(setf mh "Medium High")
(setf h "High")
(setf vh "Very High")

(setf addf-array (append
  (list (list 'VL 'L 'ML 'M 'MH 'H 'VH))
  (list (list 'L 'ML 'M 'MH 'H 'VH 'VH))
  (list (list 'ML 'M 'MH 'H 'VH 'VH 'VH))
  (list (list 'M 'MH 'H 'VH 'VH 'VH 'VH))
  (list (list 'MH 'H 'VH 'VH 'VH 'VH 'VH))
  (list (list 'H 'VH 'VH 'VH 'VH 'VH 'VH))
  (list (list 'VH 'VH 'VH 'VH 'VH 'VH 'VH))))

(setf multf-array (append
  (list (list 'VL 'VL 'VL 'VL 'VL 'VL 'VL))
  (list (list 'VL 'VL 'VL 'VL 'VL 'L 'L))
  (list (list 'VL 'VL 'VL 'L 'L 'L 'ML))
  (list (list 'VL 'VL 'L 'L 'ML 'ML 'M))
  (list (list 'VL 'VL 'L 'ML 'ML 'M 'MH))
  (list (list 'VL 'L 'L 'ML 'M 'MH 'H))
  (list (list 'VL 'L 'ML 'M 'MH 'H 'VH))))

(setf divf-array (append
  (list (list 'VH 'ML 'L 'VL 'VL 'VL 'VL))
  (list (list 'VH 'VH 'MH 'ML 'ML 'L 'L))
  (list (list 'VH 'VH 'VH 'MH 'M 'M 'ML))
  (list (list 'VH 'VH 'VH 'VH 'H 'MH 'M))
  (list (list 'VH 'VH 'VH 'VH 'VH 'H 'MH))
  (list (list 'VH 'VH 'MH 'VH 'VH 'VH 'H))
  (list (list 'VH 'VH 'VH 'VH 'VH 'VH 'VH))))

(setf subf-array (append

```

```

      (list (list 'VL 'VL 'VL 'VL 'VL 'VL 'VL))
      (list (list 'L 'VL 'VL 'VL 'VL 'VL 'VL))
      (list (list 'ML 'L 'VL 'VL 'VL 'VL 'VL))
      (list (list 'M 'ML 'L 'VL 'VL 'VL 'VL))
      (list (list 'MH 'M 'ML 'L 'VL 'VL 'VL))
      (list (list 'H 'MH 'M 'L 'L 'VL 'VL))
      (list (list 'VH 'H 'MH 'M 'M 'L 'VL))))

(defun addf (value1 value2)
  "Will return the result of 'adding' the two fuzzy values"
  (setf pos1 (position value1 fuzzy-values))
  (setf pos2 (position value2 fuzzy-values))
  (nth pos2 (nth pos1 addf-array)))

(defun subf (value1 value2)
  "Will return the result of 'subtracting' the two fuzzy values"
  (setf pos1 (position value1 fuzzy-values))
  (setf pos2 (position value2 fuzzy-values))
  (nth pos2 (nth pos1 subf-array)))

(defun multf (value1 value2)
  "Will return the result of 'multiplying' the two fuzzy values"
  (setf pos1 (position value1 fuzzy-values))
  (setf pos2 (position value2 fuzzy-values))
  (nth pos2 (nth pos1 multf-array)))

(defun divf (value1 value2)
  "Will return the result of 'dividing' the two fuzzy values"
  (setf pos1 (position value1 fuzzy-values))
  (setf pos2 (position value2 fuzzy-values))
  (nth pos2 (nth pos1 divf-array)))

```

Appendix B: Sample Program Output

B-1 Program Output - Numeric Method (50 Samples)

Results of Vulnerability Processing

Total number of Vulnerabilities Processed: 50

Statistical Results by Functional Area

	Audit	Recovery	Access	Media	O/S	Config	Docs
Influence	7.2378	7.3191	17.2539	6.0062	12.5051	12.2245	15.8863
Average	0.1448	0.1464	0.3451	0.1201	0.2501	0.2445	0.3177
Std Deviation	0.1268	0.1530	0.2478	0.1601	0.2169	0.1757	0.2192
Influence-Pct	9.23%	9.33%	22.00%	7.66%	15.94%	15.59%	20.25%

Significant Contributors to each Functional Area

Audit	V100	V125	V131	V143	V148
Recovery	V112	V110	V125	V131	V148
Access	V125	V142	V106	V101	V113
Media	V112	V106	V110	V125	V111
Operating Sys.	V125	V101	V115	V100	V144
Configuration	V143	V131	V137	V142	V125
Documentation	V127	V125	V115	V142	V131

Vulnerabilities Contributing to more than one Functional Area

Two Areas	V100	V101	V106	V110	V112	V115	V125	V131
	V142	V143	V148					
Three Areas	V125	V131	V142					
Four Areas	V125	V131						
Five Areas	V125							
Six Areas	V125							
Seven Areas	V125							

Vulnerabilities Rankings for Each Functional Area

Functional Area: AUDIT

Sigma Group: 1 ==> V100 V125 V131

Sigma Group: 2 ==> V143 V148 V115 V101 V142 V112

Sigma Group: 3 ==> V137 V141 V117 V140 V144 V129 V135 V133
V118 V124 V134 V122

Sigma Group: 4 ==> V139 V128 V113 V136 V130 V132 V127 V114
V111 V105 V108 V107 V106 V119 V110 V147
V138 V120 V145 V103 V146 V116 V102 V149
V109 V104 V123 V126

Functional Area: RECOVER

Sigma Group: 2 ==> V112

Sigma Group: 3 ==> V110 V125 V131 V148 V100 V111

Sigma Group: 4 ==> V102 V101 V103 V117 V142 V140 V144 V115
V105 V143 V135 V137

Sigma Group: 5 ==> V124 V141 V104 V133 V113 V118 V139 V122
V132 V129 V130 V107 V128 V136 V138 V121
V127 V123 V145 V114 V108 V146 V119 V134
V116 V149 V120 V147 V126 V109

Functional Area: ACCESS

Sigma Group: 1 ==> V125 V142 V106 V101 V113 V112 V143

Sigma Group: 2 ==> V115 V131 V137 V148 V144 V141 V128 V124
V100

Sigma Group: 3 ==> V117 V132 V135 V114 V110 V139 V129 V133
V119 V140 V118 V122 V111 V105 V102 V136
V108 V103 V130

Sigma Group: 4 ==> V138 V134 V116 V104 V146 V149 V107 V109
V145 V127 V147 V120 V121 V123

Functional Area: MEDIA

Sigma Group: 1 ==> V112

Sigma Group: 2 ==> V106

Sigma Group: 3 ==> V110 V125

Sigma Group: 4 ==> V111 V102 V108 V109 V119 V105 V135 V148
V107 V131 V140 V146 V100

Sigma Group: 5 ==> V117 V115 V104 V113 V101 V137 V143 V122
V133 V138 V142 V129 V127 V130 V114 V144
V121 V124 V139 V128 V134 V141 V132 V120
V118 V126 V145 V123 V136 V147 V103 V149

Functional Area: OS

Sigma Group: 1 ==>	V125	V101	V115	V100	V144			
Sigma Group: 2 ==>	V124	V131	V143	V127	V148	V142	V122	
Sigma Group: 3 ==>	V117	V137	V112	V141	V128	V140	V133	V132
	V135	V129	V109	V106	V118	V139	V119	
Sigma Group: 4 ==>	V110	V113	V111	V130	V136	V114	V108	V107
	V138	V149	V120	V146	V103	V105	V147	V102
	V145	V116	V104	V134	V126	V123		

Functional Area: CONFIGURATION

Sigma Group: 1 ==>	V143	V131	V137	V142	V125	V148	V128	V115
Sigma Group: 2 ==>	V101	V141	V113	V106	V112	V124	V100	V117
Sigma Group: 3 ==>	V135	V140	V114	V133	V129	V132	V144	V122
	V118	V139	V119	V111	V110	V103	V130	V136
	V108	V138	V146	V105	V102	V109	V107	V149
Sigma Group: 4 ==>	V104	V127	V120	V145	V147	V116	V134	V121
	V126							

Functional Area: DOCUMENTATION

Sigma Group: 1 ==>	V127	V125	V115	V142	V131	V143		
Sigma Group: 2 ==>	V101	V106	V137	V148	V113	V128	V141	V100
	V112							
Sigma Group: 3 ==>	V114	V124	V144	V117	V132	V135	V129	V140
	V133	V108	V109	V118	V122	V119	V111	V139
	V130	V110						
Sigma Group: 4 ==>	V138	V103	V116	V107	V134	V104	V136	V102
	V105	V146	V149	V120	V145	V147	V121	V123

List of Vulnerabilities

- V100 : The system does not have audit trails.
- V101 : During logon, the system does not tell the user the date and time the ID was last used
- V102 : Off-site backup does not exist for critical files
- V103 : Off-site storage is not a secure area
- V104 : Procedures have not been designed to insure only authorized personnel have access to the storage media
- V105 : Storage media reserved for use by a contractor is not clear of all classified or sensitive data
- V106 : External tape or disk identification labels do not include security classification
- V107 : Media records are not kept.
- V108 : Inventory records do not show media on hand in the library
- V109 : Guidelines and controls have not been established to designate an individual as disk manager for each computer system
- V110 : Backup is not available for power (generator or batteries)
- V111 : Backup is not available for air conditioning
- V112 : Backups of software for critical applications are not compared to working copies to detect unauthorized changes.
- V113 : Security test and evaluation is not performed prior to certification
- V114 : A contingency plan does not exist.
- V115 : System documentation does not include detailed information concerning software use for the user
- V116 : Security documentation does not include configuration management controls
- V117 : For PC (single-user) systems, files of different classifications are not limited to authorized users
- V118 : Classification of software does not take into account algorithms or processes that may be used
- V119 : Classification of media is not downgraded by reviewing all information
- V120 : For periods processing the system does not use separate copies of the operating system
- V121 : Protection of ADP magnetic storage media does not include safeguarding media according to the highest classification ever recorded
- V122 : The operating system does not automatically label all human-readable output with its sensitivity

- V123 : Main memory and storage devices are not cleared before being assigned to another individual or process.
- V124 : The operating system does not require users to identify themselves before performing any actions
- V125 : The operating system does not use a protected mechanism (e.g., passwords) to authenticate user identity
- V126 : Testing is not performed to insure that there are no ways for an unauthorized user to gain access to the system
- V127 : Documentation does not exist that describes operating system protection mechanisms
- V128 : Documentation does not exist that outlines the test plan, procedures and results for security testing
- V129 : A trusted facility manual does not include procedures for the operator to operate the facility in a secure manner
- V130 : Passwords are not randomly generated
- V131 : The password administrator's responsibilities do not include sole access to the password file
- V132 : Requirement(s) not met are that private data passwords are known only by the creator
- V133 : Password management and control does not consist of a single point of contact
- V134 : Audit trails for password distribution and change are not in existence.
- V135 : Passwords are not changed at least every three months
- V136 : Compromised or mishandled passwords are not changed at least within one working day
- V137 : Personal passwords are not deleted within three work days when a user leaves the organization
- V138 : Group passwords are not changed within three work days when a user leaves the organization.
- V139 : Resource protection measures do not include making personnel responsible for protection of government property
- V140 : The keys and combinations of the room are not restricted to a limited number of holders
- V141 : The keys and combinations of the room are not changed on a regular basis
- V142 : The doors and gates of the room are not kept closed at all times
- V143 : The windows of the room are not kept closed at all times
- V144 : Systems are not located such that access is controlled
- V145 : Controls for small computer users do not include cold-booting at the start of each session if classified

- V146 : Diskettes are not write-protected when it is appropriate to do so
- V147 : Users do not know they should not use personally owned computers or systems at home for Air Force business
- V148 : Changes to software are not documented
- V149 : Access to utility software is not limited to specifically identified personnel.

B-2 Program Output - Fuzzy Method (50 Samples)

Results of Vulnerability Processing

Total number of Vulnerabilities Processed: 50

Statistical Results by Functional Area

	VH	H	MH	M	ML	L	VL
Audit	0	0	0	2	5	14	29
Recovery	1	0	0	1	6	12	30
Access	3	4	6	4	5	18	10
Media	1	0	1	1	2	11	34
Operating Sys.	1	2	3	6	4	12	22
Configuration	0	0	4	6	7	14	19
Documentation	1	4	5	5	8	18	9

Significant Contributors to each Functional Area

Audit	F125	F100	F101	F112	F131	F143	F148	
Recovery	F112	F110	F101	F125	F100	F131	F148	F111
Access	F101	F125	F142	F112	F106	F113	F143	
Media	F112	F106	F110	F125	F111			
Operating Sys.	F101	F125	F115	F100	F124	F144		
Configuration	F142	F131	F143	F137	F101	F112	F125	F115
	F148	F128						
Documentation	F127	F125	F142	F115	F143			

Vulnerabilities Contributing to more than one Functional Area

Two Areas	F100	F101	F106	F110	F111	F112	F115	F125
	F131	F142	F143	F148				
Three Areas	F100	F101	F112	F115	F125	F131	F142	F143
	F148							
Four Areas	F101	F112	F125	F143				
Five Areas	F101	F112	F125					
Six Areas	F125							
Seven Areas	F125							

Vulnerabilities Rankings for Each Functional Area

Functional Area: AUDIT

Importance: M ==> F125 F100

Importance: ML ==> F101 F112 F131 F143 F148

Importance: L ==> F122 F142 F115 F124 F144 F137 F141 F117
F133 F135 F118 F129 F134 F140

Importance: VL ==> F127 F106 F113 F108 F128 F132 F110 F136
F139 F105 F111 F114 F119 F102 F109 F130
F107 F103 F138 F146 F104 F116 F120 F147
F123 F126 F145 F149 F121

Functional Area: RECOVER

Importance: VH ==> F112

Importance: M ==> F110

Importance: ML ==> F101 F125 F100 F131 F148 F111

Importance: L ==> F142 F115 F124 F143 F144 F117 F135 F102
F105 F103 F104 F140

Importance: VL ==> F122 F127 F106 F113 F137 F141 F132 F133
F108 F128 F118 F129 F139 F109 F114 F119
F136 F107 F130 F138 F146 F116 F134 F121
F123 F126 F145 F149 F120 F147

Functional Area: ACCESS

Importance: VH ==> F101 F125 F142

Importance: H ==> F112 F106 F113 F143

Importance: MH ==> F115 F131 F144 F137 F141 F148

Importance: M ==> F100 F124 F117 F128

Importance: ML ==> F132 F135 F133 F118 F119

Importance: L ==> F122 F110 F108 F114 F129 F139 F105 F111
F136 F102 F130 F138 F140 F103 F104 F116
F134 F146

Importance: VL ==> F127 F109 F107 F121 F123 F145 F149 F120
F147 F126

Functional Area: MEDIA

Importance: VH ==> F112

Importance: MH ==> F106

Importance: M ==> F110

Importance: ML ==> F125 F111

Importance: L ==> F100 F131 F108 F117 F135 F109 F102 F105
F119 F146 F107

Importance: VL ==> F101 F122 F127 F142 F113 F115 F124 F143
F144 F148 F137 F141 F128 F132 F133 F114
F118 F129 F136 F139 F104 F140 F130 F138
F103 F116 F134 F121 F123 F126 F120 F145
F147 F149

Functional Area: OS

Importance: VH ==> F101

Importance: H ==> F125 F115

Importance: MH ==> F100 F124 F144

Importance: M ==> F122 F127 F142 F131 F143 F148

Importance: ML ==> F112 F137 F141 F117

Importance: L ==> F106 F113 F128 F132 F133 F135 F109 F118
F129 F119 F139 F140

Importance: VL ==> F108 F110 F105 F111 F136 F102 F114 F130
F107 F138 F146 F103 F104 F116 F134 F120
F149 F126 F147 F123 F145 F121

Functional Area: CONFIGURATION

Importance: MH ==> F142 F131 F143 F137

Importance: M ==> F101 F112 F125 F115 F148 F128

Importance: ML ==> F100 F106 F113 F124 F141 F117 F135

Importance: L ==> F122 F144 F132 F133 F108 F110 F114 F118
F129 F111 F119 F139 F140 F130

Importance: VL ==> F127 F102 F105 F109 F136 F103 F104 F138
F146 F107 F116 F134 F126 F149 F120 F121
F123 F145 F147

Functional Area: DOCUMENTATION

Importance: VH ==> F127

Importance: H ==> F125 F142 F115 F143

Importance: MH ==> F101 F106 F131 F137 F148

Importance: M ==> F112 F100 F113 F141 F128

Importance: ML ==> F124 F144 F117 F132 F135 F133 F114 F129

Importance: L ==> F122 F108 F110 F109 F118 F111 F119 F105
F136 F139 F104 F130 F138 F140 F107 F116
F134 F146

Importance: VL ==> F102 F103 F149 F120 F121 F123 F145 F147
F126

List of Vulnerabilities

- F100 : The system does not have audit trails.
- F101 : During logon, the system does not tell the user the date and time the ID was last used
- F102 : Off-site backup does not exist for critical files
- F103 : Off-site storage is not a secure area
- F104 : Procedures have not been designed to insure only authorized personnel have access to the storage media
- F105 : Storage media reserved for use by a contractor is not clear of all classified or sensitive data
- F106 : External tape or disk identification labels do not include security classification
- F107 : Media records are not kept.
- F108 : Inventory records do not show media on hand in the library
- F109 : Guidelines and controls have not been established to designate an individual as disk manager for each computer system
- F110 : Backup is not available for power (generator or batteries)
- F111 : Backup is not available for air conditioning
- F112 : Backups of software for critical applications are not compared to working copies to detect unauthorized changes.
- F113 : Security test and evaluation is not performed prior to certification
- F114 : A contingency plan does not exist.
- F115 : System documentation does not include detailed information concerning software use for the user
- F116 : Security documentation does not include configuration management controls
- F117 : For PC (single-user) systems, files of different classifications are not limited to authorized users
- F118 : Classification of software does not take into account algorithms or processes that may be used
- F119 : Classification of media is not downgraded by reviewing all information
- F120 : For periods processing the system does not use separate copies of the operating system
- F121 : Protection of ADP magnetic storage media does not include safeguarding media according to the highest classification ever recorded
- F122 : The operating system does not automatically label all human-readable output with its sensitivity

- F123 : Main memory and storage devices are not cleared before being assigned to another individual or process.
- F124 : The operating system does not require users to identify themselves before performing any actions
- F125 : The operating system does not use a protected mechanism (e.g., passwords) to authenticate user identity
- F126 : Testing is not performed to insure that there are no ways for an unauthorized user to gain access to the system
- F127 : Documentation does not exist that describes operating system protection mechanisms
- F128 : Documentation does not exist that outlines the test plan, procedures and results for security testing
- F129 : A trusted facility manual does not include procedures for the operator to operate the facility in a secure manner
- F130 : Passwords are not randomly generated
- F131 : The password administrator's responsibilities do not include sole access to the password file
- F132 : Requirement(s) not met are that private data passwords are known only by the creator
- F133 : Password management and control does not consist of a single point of contact
- F134 : Audit trails for password distribution and change are not in existence.
- F135 : Passwords are not changed at least every three months
- F136 : Compromised or mishandled passwords are not changed at least within one working day
- F137 : Personal passwords are not deleted within three work days when a user leaves the organization
- F138 : Group passwords are not changed within three work days when a user leaves the organization.
- F139 : Resource protection measures do not include making personnel responsible for protection of government property
- F140 : The keys and combinations of the room are not restricted to a limited number of holders
- F141 : The keys and combinations of the room are not changed on a regular basis
- F142 : The doors and gates of the room are not kept closed at all times
- F143 : The windows of the room are not kept closed at all times
- F144 : Systems are not located such that access is controlled
- F145 : Controls for small computer users do not include cold-booting at the start of each session if classified

- F146 : Diskettes are not write-protected when it is appropriate to do so
- F147 : Users do not know they should not use personally owned computers or systems at home for Air Force business
- F148 : Changes to software are not documented
- F149 : Access to utility software is not limited to specifically identified personnel.

Bibliography

- Bacchus, Fahiem. Representing and Reasoning with Probabilistic Knowledge. Cambridge MA: The MIT Press, 1990.
- Bureau of the Census. Statistical Abstract of the United States 1992 (112th Edition). Washington: GPO, 1993.
- Carroll, John M. Managing Risk: A Computer-Aided Strategy. Boston, MA: Butterworth Publishers, 1984.
- Dempster, A. P. "Upper and Lower Probabilities Induced by Multivalued Mappings," Annals of Mathematical Statistics. 38:325-329. 1967.
- Department of the Air Force (DAF). Air Force Systems Security Memorandum: Network Risk Analysis Guide. AFSSM 5022. Washington: HQ USAF, 1 March 1993a.
- Department of the Air Force (DAF). Air Force Systems Security Memorandum: Risk Analysis. AFSSM 5018. Washington: HQ USAF, 1 February 1993b.
- Department of the Air Force (DAF). Security: Computer Security Policy. AFR 205-16. Washington: HQ USAF, 28 April 1989.
- Dillard, R. A. Statistical Decision Making with Uncertain and Conflicting Data: Technical Report 1451. San Diego: Naval Ocean Systems Center, September 1991 (AD-A243853).
- Dontas, Kejitan. An Implementation of the Collins-Michalski Core Theory of Plausible Reasoning. MS Thesis (Draft). Knoxville, TN: Department of Computer Science, University of Tennessee, September 1987.
- Hoffman, Lance J. and Lee A. Neitzel. "Inexact Analysis of Risk," Proceedings of the International Conference on Cybernetics and Society. 366-372. New York, NY: Institute of Electrical and Electronics Engineers, 1980.
- Kaufmann, Arnold and Madan M. Gupta. Introduction to Fuzzy Arithmetic: Theory and Applications. New York, NY: Van Nostrand Reinhold Company, 1985.

- Giarratano, Joseph and Gary Riley. Expert Systems: Principles and Programming. Boston: PWS-KENT Publishing Company, 1989.
- Lukasiewicz, J. "Many-valued Systems of Propositional Logic," Polish Logic. Oxford University Press, 1967.
- Martin-Lof. "Constructive Mathematics and Computer Programming," Methodology and Philosophy of Science IV. 153-175. Amsterdam: North Holland Publishing Company, 1982.
- McDermott, D. "A Temporal Logic for Reasoning about Plans and Actions," Cognitive Science. 6:101-155. 1982.
- Michalski, Ryszard S. Two-tiered Concept meaning, inferential matching and cohesiveness. Invited paper for the Allerton Conference on Analogy and Similarity, 1986.
- Michalski, Ryszard S. and Patrick H. Winston. "Variable Precision Logic," Artificial Intelligence. 29:121-146. 1986.
- Nagy, T. J. and L. J. Hoffman. Exploratory Evaluation of the Accuracy of Linguistic vs. Numeric Risk Assessment of Computer Security. Technical Report GWU-IIST-81-07. Computer Security Research Group, The George Washington University. May 1981.
- Negoita, Constantin Virgil. Expert Systems and Fuzzy Systems. Menlo Park CA: The Benjamin/Cummings Publishing Company, Inc., 1985.
- Oliver, R. M. and J. Q. Smith, eds. Influence Diagrams, Belief Nets and Decision Analysis. Chichester, England: John Wiley & Sons, 1990.
- Pearl, Judea. Probabilistic Reasoning in Intelligent Systems. Palo Alto: Morgan Kaufmann, 1988.
- Podell, Harold J. and Marshall D. Abrams. "A Computer Security Glossary for the Advanced Practitioner," Computer Security Journal, Volume IV, Number 1. 69-88. Northborough, MA: Computer Security Institute, 1986.
- Quinlan, J. R. Inferno: A Cautious Approach to Uncertain Inference. RAND Note N-1898-RC. Santa Monica: RAND Corporation, 1982.

- Reiter, R. "Logic for Default Reasoning," Artificial Intelligence. 13:1-132. 1980.
- Rich, Elaine and Kevin Knight. Artificial Intelligence (Second Edition). New York: McGraw-Hill, Inc., 1991.
- Sanchez, E. and Zadeh, L. A. ed. Approximate Reasoning in Intelligent Systems. Decision and Control. Oxford, England: Pergamon Press, 1987.
- Scheaffer, Richard L. and James T. McClave. Probability and Statistics for Engineers. Boston, MA: Duxbury Press, 1986.
- Schmucker, Kurt J. Fuzzy Sets. Natural Language Computations. and Risk Analysis. Rockville MD: Computer Science Press, 1984.
- Shafer, Glenn and Roger Logan. "Implementing Dempster's Rule for Hierarchical Evidence," Artificial Intelligence. 33(3):271-298. November 1987.
- Trident Data Systems. Automated Risk Evaluation System (ARES) Maintenance Plan. Technical plan to Air Force Cryptological Support Center, Kelly AFB, TX, 12 April 1993.
- Wong, Kenneth K. Risk Analysis and Control. Oxford, England: NCC Publications, 1977.
- Wood, Charles C. and others. Computer Security: A Comprehensive Controls Checklist. New York, NY: John Wiley & Sons, 1987.
- Yager, Ronald. "Using Approximate Reasoning to Represent Default Knowledge," Artificial Intelligence. 31:99-112. 1987.
- Zadeh, Lofti A. "Fuzzy Sets," Information and Control, vol. 8. 338-353. New York NY: Academic Press, 1965.
- Zadeh, Lofti A. and Janusz Kacprzyk. ed. Fuzzy Logic for the Management of Uncertainty. New York, NY: John Wiley & Sons, Inc., 1992.

VITA

Captain Richard W. Fleming was born on 18 August 1963 in Odessa, Texas. He graduated from Odessa Senior High School in Odessa, Texas in 1981 and enlisted in the U. S. Air Force that same year. After serving a tour as a computer operations specialist, he applied and was selected for the Airman's Education and Commissioning Program (AECPP). He attend New Mexico State University in Las Cruces, New Mexico and received his Bachelor of Science in Electrical Engineering in May 1988. Upon graduation, he attend the Officer Training School, located at Lackland AFB, Texas and received a reserve commission in the USAF. His first assignment following commissioning was to Edwards AFB, California. There he served with the B-2 Combined Test Force (CTF) while on loan from the 6521 Range Squadron. He began as a computer systems analyst where he designed, developed, tested, and evaluated hardware and software systems for the B-2 flight test program. He was then chosen to serve as the branch chief for the computer operations branch of the B-2 CTF. There he was responsible for insuring the readiness and availability of real time computer support for the B-2 flight test missions. He served in that position until entering the School of Engineering, Air Force Institute of Technology, in May 1992. Following completion of his Master in Computer Engineering Degree at AFIT, Captain Fleming was assigned to the Information Systems Group, part of the Air Intelligence Agency, located at Kelly AFB, Texas.

Permanent Address
Box 103
Goldsmith, TX 79741

REPORT DOCUMENTATION PAGEForm Approved
OMB No 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE December 1993	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE VULNERABILITY ASSESSMENT USING A FUZZY LOGIC BASED METHOD			5. FUNDING NUMBERS	
6. AUTHOR(S) Richard W. Fleming, Captain, USAF				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, WPAFB OH 45433-6583			8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GCE/ENG/93D-03	
9. SPONSORING MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Cryptological Support Center AFCSC/SROV Kelly AFB, TX 78243-5000			10. SPONSORING MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION AVAILABILITY STATEMENT Distribution Unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This thesis demonstrates the feasibility of using qualitative analysis methods to evaluate computer security vulnerabilities. Although many risk analysis systems exist, few provide for the adequate analysis of identified vulnerabilities. While the main focus of this thesis is to evaluate computer security vulnerabilities, the methods involved have application in other areas requiring evaluation using qualitative methods. It is proposed, and demonstrated by this thesis, that the use of qualitative analysis using linguistic variables to describe the impact of computer security vulnerabilities is not only feasible, but intrinsically easier to understand and use than quantitative methods.				
14. SUBJECT TERMS ARTIFICIAL INTELLIGENCE; VULNERABILITY ASSESSMENT; FUZZY LOGIC; COMPUTER SECURITY			15. NUMBER OF PAGES 101	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	